THE IMPLEMENTATION AND ANALYSIS
OF
WIDE DYNAMIC RANGE IMAGING METHODS

by

WAZIRAH BINTI MD ESA

DISSERTATION

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

THE IMPLEMENTATION AND ANALYSIS
OF
WIDE DYNAMIC RANGE IMAGING METHODS

by

Wazirah binti Md Esa

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

_____

(Dr. Aamir Saeed Malik)
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

December 2010

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

(Wazirah binti Md Esa)

ABSTRACT

This dissertation presents the implementation and analysis of wide dynamic range (WDR) imaging methods. It covers the study of wide dynamic range imaging, its applications and techniques. In Literature Review part, some of the literatures published by WDR imaging researchers which are useful for this project completion are summarized. In Methodology part, the implementation of the methods in WDR imaging is presented. The author has successfully implemented WDR image generation, tone-mapping and their combinations. In Results and Discussion part, some measures, analysis, and observations and for each output image resulted from those methods are explained. Finally, in Conclusion and Recommendation part, ideas and project results are wrapped up.

ACKNOWLEDGEMENTS

There are many people who deserve my huge amounts of thanks and credit for their help along the way. First and foremost, I express my sincere thanks to my supervisor, Dr. Aamir Saeed Malik, for his support and guidance through my project completion. Without the opportunity and trust he gave me, this dissertation would have never been possible.

My sincere thanks also go to my colleagues and best friends, Fara Rozana, Nadia Asyikin, Alif Imran, Razan and Mohamad Zikir, for the friendship, motivation and healthy competition. Also to the post-graduate students, my course-mates and my friends who cooperatively involved during the evaluation of my project results. The list of other people who also deserve my thanks is too long to completely write down all of them here.

Lastly and certainly, my parents and my siblings deserve my gratitude. They have always be my inspiration in my years of study. I am indebted to them for their constant love, encouragement and understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

WDR: Wide Dynamic Range

HDR: High Dynamic Range

LDR: Low Dynamic Range

CHAPTER 1
INTRODUCTION

1.1 Background

Wide Dynamic Range (WDR) or well-known as High Dynamic Range (HDR) imaging is a solution for improving images captured from scenes which have both bright and dark areas. It covers the processes of image capture and visualization. If compared to low dynamic range (LDR) imaging, the WDR imaging performs better in exposing hidden (over- and under-exposed) details in the scenes. Usually, WDR images are acquired from scenes which have wide range of light intensity variations [1].

Real world scenes contain vast range of luminance (light intensities) which varies from dim starlight to direct sunlight [2 – 5]. However, common cameras are only capable of capturing images of 8-bit color depth. 8-bit color depth means that there are 8 bits per pixel for each of red, green and blue channels. Thus, there are 256 luminance levels per color channel [6]. Figure 1 shows range of luminance of real-world.

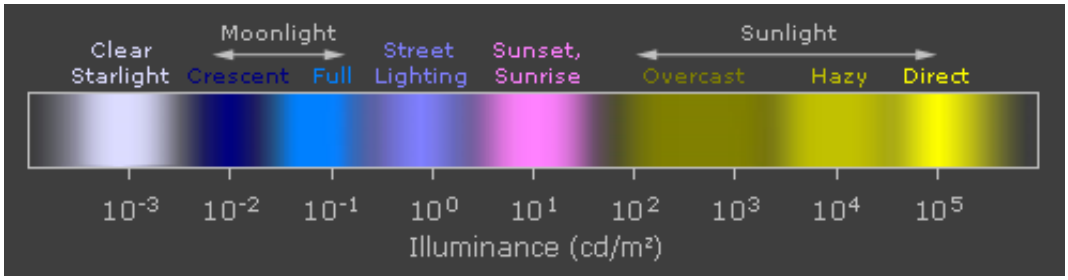


Figure 1 Range of Luminance (or Illuminance) measured in cd/m^2 [7]

Generally, dynamic range is the ratio between maximum and minimum light intensities. To cover dynamic range of real world scenes, 256 luminance levels per color channel captured by conventional cameras are not enough especially when both bright and dark areas are involved in the same scene. Therefore, during image capture, dynamic range of original scene is reduced due to this limitation. This scenario is demonstrated in Figure 2. As a consequence, conventional image capture has caused lost of details in dark (under-exposed) and bright (over-exposed) areas [3, 8].

For WDR image capture, WDR cameras which have wide or high dynamic range are capable of covering details of a scene which have both bright and dark areas. Unfortunately, they are expensive, about 50,000 US dollars [3]. Therefore, in order to obtain a WDR image of such scene by using common cameras, multiple LDR images with different exposure settings are fused or combined to extend or increase the dynamic range [9 - 11].

Figure 2 Reduction of Dynamic Range of Original Scene during Image Capture [7]

Apart from WDR image capture, it is a common need to display WDR images on display devices. However, currently, 8-bit low dynamic range (LDR) display devices are the most widely used and their dynamic range is approximately 100:1 which is much lower than dynamic range of the WDR images [3, 12 - 14]. Since WDR images have higher color depth, dynamic range of WDR images must be reduced so that it can be displayed in some appreciable way on LDR display devices. In the same time, we also have to preserve details,

brightness, contrast impression and color appearance of the images [11]. This process is called WDR visualization.

1.2 Problem Statement

a) To generate WDR images
b) To prepare the generated WDR images so that they can be displayed properly on display devices with low dynamic range

1.3 Objectives and Scope of Study

a) To implement two methods of WDR image generation, five methods of WDR visualization and their combinations
b) To analyze output images resulted from the implemented WDR image generation and visualization methods through both objective and subjective measures

CHAPTER 2
LITERATURE REVIEW

2.1 Wide Dynamic Range (WDR)

In digital image processing, dynamic range is defined as the ratio of the maximum measurable intensity level to the minimum detectable intensity level in the system (scene, image and display device) [15, 16]. When dynamic range of a scene or an image is wide or high, they are considered as high contrast. Conversely, when they have low dynamic range, they are considered as low contrast [15]. For a display device, its dynamic range is the ratio of the maximum light intensitiy to the minimum light intensity emitted from the screen [16]. WDR image usually has 32 bits per color channel while LDR image has up to 8 bits per color channel [16].

2.2 WDR Image Generation

Most of common digital cameras can only capture limited dynamic range because every sensor has its own response property [17]. This means, for the case of very high contrast in a real world scene, they can capture only part of dynamic range of the scene. Thus, due to the loss during image capture itself, it is not easy to display all details apparently [18].

Most researchers solve the problem by taking multiple LDR images of the same scene with different exposure settings and then combining them into a single WDR one [6, 9-11]. The purpose of this approach is to gain more light intensities information and to extend dynamic range so that resulting WDR image keeps more visible details from entire scene's dynamic range [10, 11].

2.2.1 MATLAB "makehdr" Function

"makehdr" is a function available in MATLAB that creates a WDR image from a series of differently exposed LDR images (Please refer to **Appendix A**). Middle exposure between the brightest and the darkest images is used as base exposure for calculations [19]. It is assumed that capturing device is perfectly linear. Each exposure is put into the same unit of measurement by dividing each pixel with image's exposure time. Then, corresponding pixels of all LDR images (excluding over- and under-exposed pixels) are averaged to produce a WDR image [6].

2.2.2 P. Debevec and J. Malik

In [9] an algorithm is developed to generate a single radiance map (WDR image) from multiple LDR images with different exposure settings. The differently exposed LDR images are used to recover camera response function up to factor of scale. Usually, in camera response curve, it is observed that there are small response at no exposure and saturation at high exposure. The recovered camera response function is constructed based on exploitation of reciprocity, which is a physical property of imaging system. It is solved in a least-squared error sense. It also incorporates a weight function which is a simple hat function, to give more weight to smoothness and to fit more details toward the middle of curve. Once the camera response function is determined, the algorithm can combine the LDR images into a single radiance map (WDR image). This algorithm has been able to help many WDR imaging researchers in their works including [4, 10-11, 17-18, 20-23].

2.3 WDR Image Visualization (Tone-Mapping)

Previously, it has been discussed on how to generate the WDR images by combining multiple LDR images of the same scene with different exposure settings. However, most of common display devices are of 8-bit contrast ratio [3, 12 - 14]. They are are considered as low dynamic range (LDR) display devices. As a result, the WDR images having the higher color depth could not be displayed on those LDR display devices directly.

To handle this problem, there are two choices available. The first method is to buy a very expensive WDR display to display the WDR image. The second method is to prepare and reproduce the WDR image before displaying it on LDR display devices [3, 6]. The second method is called tone-mapping or tone-reproduction. Tone-mapping is the process of reducing dynamic range of WDR image to a lower one so that it can be displayed on LDR display devices [3, 6]. It is also can be defined as a method to map scene luminance to display luminance to generate a reliable image [5]. [5] had introduced global and local tone-mapping operators which were inspired by Zone System developed by Ansel Adams [24 - 26] and dodging-and-burning printing techniques. [5] also had been used widely as the main references in [3, 12-13, 27].

Commonly, tone mapping operators are classified into four categories - global, local, frequency domain, and gradient domain operators. The classification is done to differentiate operators roughly based on how light reflects from a diffuse surface from operators working directly on the pixels. A global operator uses the entire image as the neighborhood for each pixel. Therefore, same compression curve is used to compress each pixel. On the other hand, a local operator uses value of a pixel itself and values of its local neighborhood to compute its adaptation level. Then, different compression curve will be generated for each pixel. Therefore, a bright pixel in a dark region will be considered differently than a bright pixel in a bright region. Same goes for dark pixel in dark and bright region. When an operator reduces the dynamic range of image component selectively based on their spatial frequency, it is known as a frequency

domain operator. Lastly, when an operator alters the derivative of an image to achieve dynamic range reduction, it is called gradient domain operator. [6]

[25 - 27] introduced an approach called Zone System. The Zone System is a method used to determine how scene luminance is mapped to a set of print zones. In Zone System, print zones consist of Roman numerals which are related to approximated scene luminance and approximated reflectance of a print. There are eleven zones which range from pure black (Zone 0) to pure white (Zone X). If dynamic range of a scene is within the nine zones (middle grey) where pure black and pure white are excluded, all details of the scene can be appropriately mapped. Else, some of the details are mapped to pure black or pure white. The losses of details in the scene could be either desirable or objectionable. To overcome the objectionable losses of details, dodging-and-burning printing technique is used. In addition, the system introduced terminology of 'key of the scene' which subjectively determines whether a scene is light, normal or dark.

2.3.1 Logarithmic

Logarithmic tone-mapping is one of the simplest tone-mapping methods. It is useful to map high intensity range into lower one since logarithm is a compressive function for values larger than 1. This tone-mapping is suitable for medium dynamic range (MDR) images (i.e., images with a dynamic range somewhat higher than can be displayed by LDR display devices). Usually, it produces dull images. [6]

2.3.2 Exponential

Exponential tone-mapping is also one of the simplest tone-mapping methods. It is useful to compress wide intensity range since the function is bound between 0 (black) and 1 (white). This tone-mapping is also suitable for medium dynamic range (MDR) images. Usually, it produces light images. [6]

2.3.3 Reinhard et Global Tone-Mapping Operator

Key of the scene (measure of brightness or darkness) is approximated using average scene luminance. Then, the scaled scene luminance is calculated based on the obtained values of average scene luminance and key of the scene. To take high luminance values into account, the scaled luminance is rescaled. This method is claimed to sufficiently preserve details in low contrast and in the same time compressing high luminance values to displayable range. However, there are losses of details for very wide dynamic range images. [4,6]

2.3.4 Reinhard et al Local Tone-Mapping Operator

The proposed local operator is similar to process of dodging-and-burning which applied over region bounded by large contrast. A traditional center-surround is used to identify the region with large contrast. A Gaussian-weighted average for a pixel (the center) is compared with a Gaussian-weighted average over a larger portion (the surround). Both center and surround Gaussians are centered over the same pixel. [4,6]

The difference between these two Gaussians will be close to 0 if there is no large contrast in the pixel's neighborhood. However, the difference will be significant if there is contrast edge exists at the surround Gaussian but not at the center Gaussian. This operator also has a built-in sharpening parameter. Next, largest scale for which the difference of Gaussians remains below the threshold is calculated to identify the largest area that has low contrast for a given pixel. The center Gaussian at this scale may be considered as local average of the image. [4,6]

It is claimed that this operator increases contrast of bright regions with dark pixels and compresses bright pixels in a relatively dark regions in minimal way. [4,6]

2.3.5 Garrett et al Method

An image appearance model, iCAM was proposed as one of methods to tone-map the HDR images. It was designed to predict the perceptual response towards spatially complex stimuli. However, actually, it was not designed solely as tone-mapping algorithm, but rather as color predictor of overall color appearance. [6,28]

Firstly, the RGB input image is transformed into CIE 1931 XYZ tristimulus values so that it is in device independent coordinates. Next step is to apply chromatic and luminance adaptation transform which serve as local adaptation of the WDR scenes and global whitepoint shift towards CIE D65. The chromatic adaptation transform is identical to that of CIECAM (2002) [29].

To perform local adaptation, a low-pass version of the image is used as the adapting whitepoint. On the other hand, to perform global adaptation, the whitepoint is set to match scenes under D65 (average daylight) since later IPT transform is designed only for D65 scenes. Basically, chromatic adaptation transform consists of three parameters. They are the amount of blurring in the low-pass image, the degree of adaptation, and the choice between chromatic and luminance adaptation. [6,28]

After that, the local contrast of the image is adjusted through a series of local tone-mapping curves. The curves represent changes in local contrast, as a result of changes on localized surround and luminance. Next, the manipulated XYZ units are converted into LMS cone responses. Then, the LMS cone responses are brought into the IPT appearance space to calculate appearance correlation such as brightness, chroma and hue. Finally, the LMS cone responses are converted back into XYZ units and RGB color space before scaling the image for display. [6,28]

2.3.6 MATLAB "tonemap"Function

"tonemap" is a function available in MATLAB that tone-maps WDR image so that it can be displayed properly on LDR display devices [30]. It converts WDR image into an LDR image and followed by some manipulations in sRGB, L*a*b* and XYZ colorspaces (Please refer to **Appendix J**).

CHAPTER 3
METHODOLOGY / PROJECT WORK

3.1 Procedure Identification

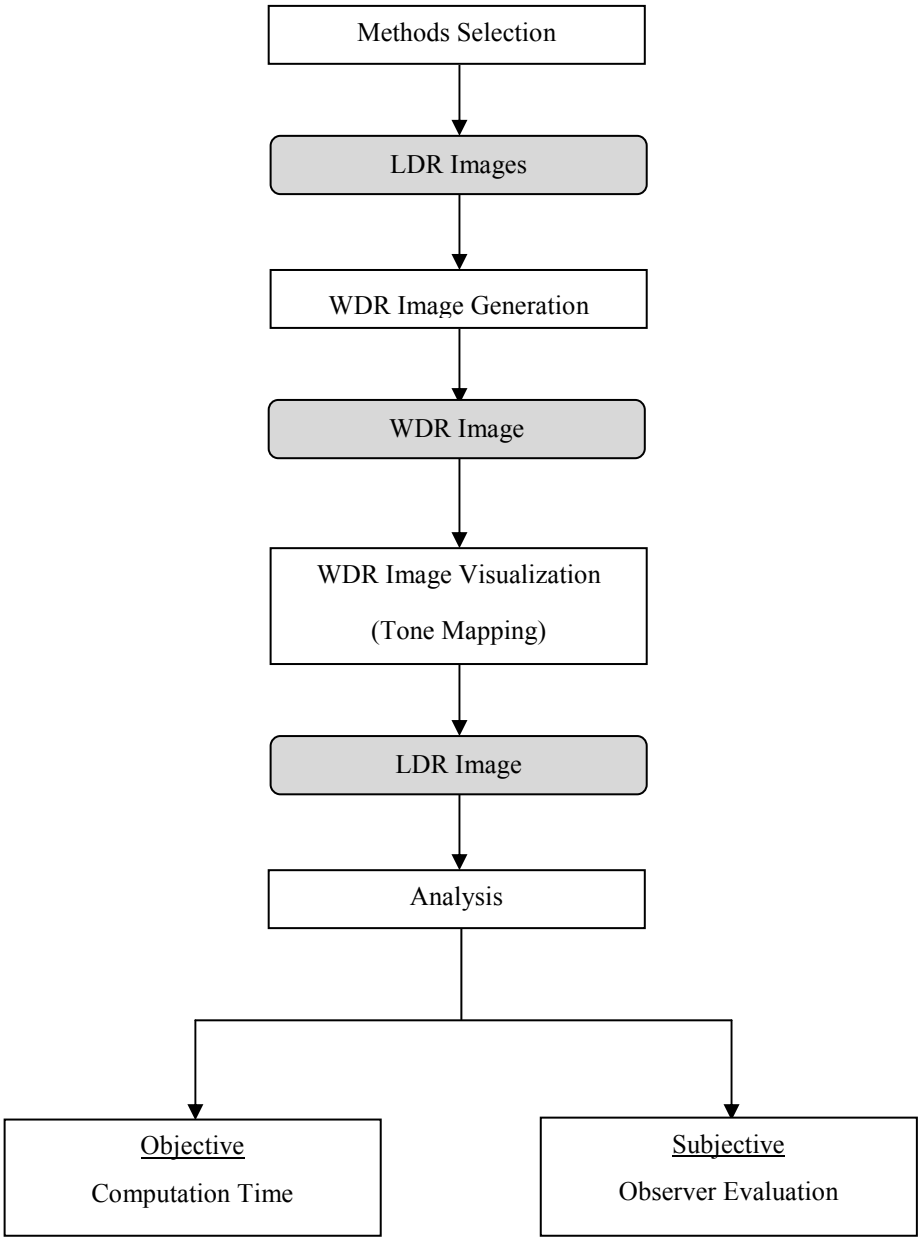


Figure 3 Flow Chart of the Project Methodology

3.2 Software

3.2.1 MATLAB

Selected algorithms were implemented using MATLAB. All programming, mathematical operations and image processing were done using this software.

3.2.2 Photomatix

In order to compare output images of the implemented algorithms with that of popular WDR software, Photomatix [31] is used. Photomatix is an innovative software used to generate and tone-map WDR images. It is developed by HDRsoft, a company which is previously known as MultimediaPhoto. Photomatix was released in February 2003. It is a succession of a research project started in July 2002 which was awarded a grant from the French Ministry of Research, as laureate of their national contest in 2003. One of the two owners and founders is a professional photographer who has over 25 years' experience. This software is widely used by WDR photographer all over the world. Their collection can be viewed at [32]. This software has three main tools. They are listed as follows:

a) WDR image generator

It generates a WDR image from LDR images with different exposure settings. The LDR images must contain EXIF (exchangeable image file format) information. Else, exposure setting of each source image must be provided manually. As shown in Figure 4, this tool offers images alignment, and reduction of chromatic aberrations, noise and ghosting artifacts. Throughout the project progression, source images are aligned by correcting horizontal and vertical shifts and recommended setting for tone curve is used.

b) Details enhancer (Tone-mapping)

As shown in Figure 5, by using this tool, one can control strength of the tone-mapping operator, color saturation, luminosity, microcontrast, and smoothing. Also, there are additional tone, color and miscellaneous settings. Throughout the project progression, default setting was used.

c) Tone compressor (Tone-mapping)

This tool allows one to control level of brightness, tonal range compression, contrast adaptation, temperature and saturation. Also, as shown in Figure 6, percentage of whitepoint and blackpoint can be adjusted. Throughout the project progression, different settings are used as the following:

Table 1 Photomatix Tone Compressor Setting

WDR Image Generation Method	Setting		WDR Image
	Brightness	Tonal Range Compression	
Standard WDR Image	Default	Default	All
MATLAB "makehdr"	-10	5.0	All, except book.hdr, lab1.hdr, and lab2.hdr
P. Debevec and J. Malik Method	-7.5	2.5	All, except book.hdr, lab1.hdr, and lab2.hdr
Photomatix	Default	Default	All

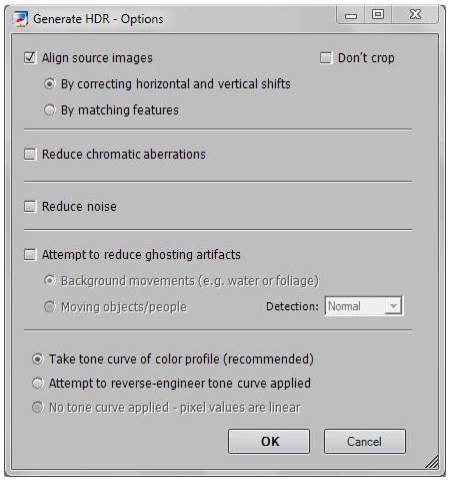


Figure 4 Options for WDR Image Generation

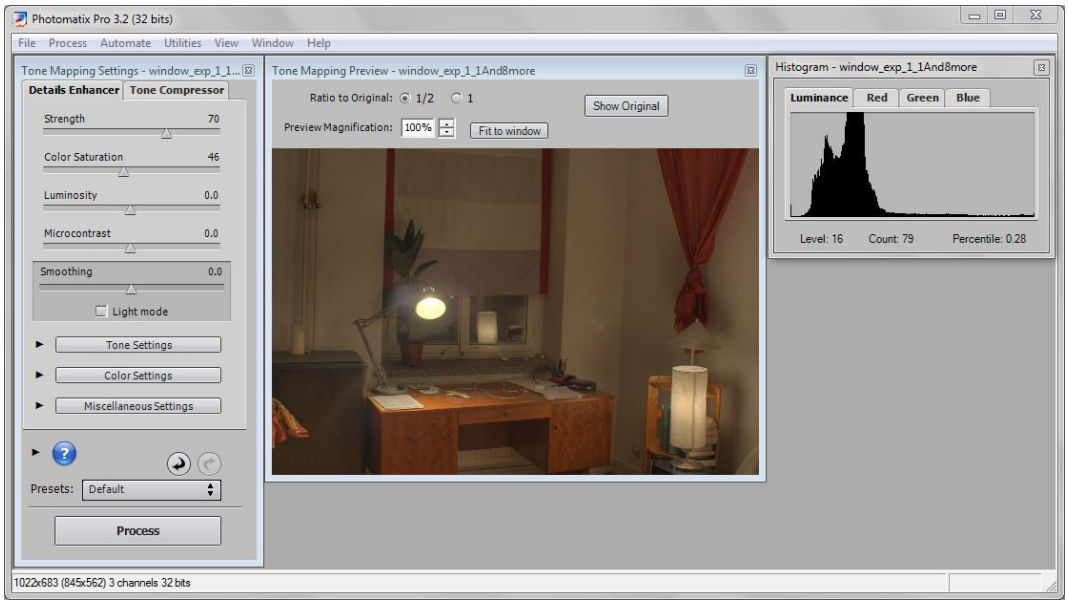


Figure 5 Details Enhancer Window

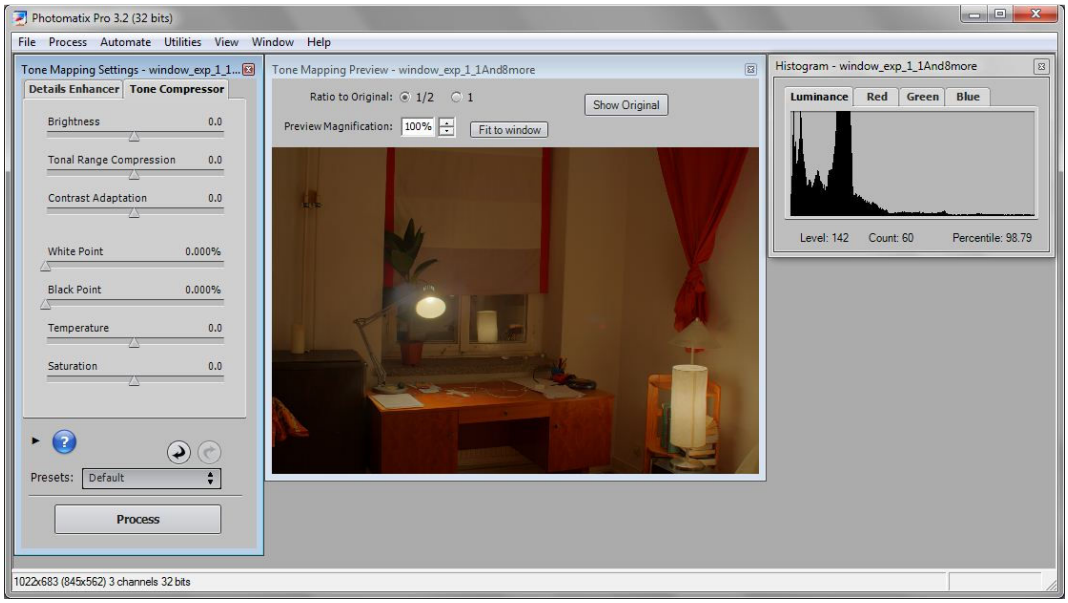


Figure 6 Tone Compressor Window

3.3 Methods Selection

3.3.1 WDR Image Generation

The implemented WDR generation methods are:

1. MATLAB “makehdr” Function
2. P. Debevec and J. Malik
   - Based on original implementation of [35]
3. Photomatix WDR Generator

All source codes of the implemented WDR image generation methods can be referred at **Appendix B** until **Appendix H**.

3.3.2 WDR Visualization (Tone-Mapping)

The implemented WDR tone-mapping methods are:

1. Logarithmic
2. Modified Logarithmic
3. Exponential
4. Modified Exponential
5. Reinhard et al Global Operator
   - Based on original implementation of [35]
6. Reinhard et al Local Operator
   - Based on original implementation of [35]
7. Garrett et al
   - Based on original implementation of [36]
8. MATLAB “tonemap” Function
9. Photomatix Details Enhancer
10. Photomatix Tone Compressor

All source codes of the implemented tone-mapping methods can be referred at **Appendix K** until **Appendix AA**.

3.3.3 Combinations of WDR Generation and Visualization (Tone-Mapping)

All WDR generation and WDR tone-mapping methods are combined as tabulated in Table 2 below. √ symbol indicates that a particular combination of WDR generation and tone-mapping is implemented.

Table 2 Implemented Combination of WDR Generation and Tone-Mapping

Tone-mapping Methods	WDR Generation Methods		
	MATLAB "makehdr" Function	P. Debevec and J. Malik	Photomatix WDR Generator
Logarithmic	√	√	√
Modified Logarithmic	√	√	√
Exponential	√	√	√
Modified Exponential	√	√	√
Reinhard et al (Global Operator)	√	√	√
Reinhard et al (Local Operator)	√	√	√
Garrett et al	√	√	√
MATLAB "tonemap" Function	√	√	√
Photomatix Detail Enhancer	√	√	√
Photomatix Tone Compressor	√	√	√

3.4 LDR Source Images

All chosen WDR generation methods are applied to the set of LDR source images listed in the following Table 3. The images can be referred at **Appendix BB** until **Appendix EE**.

Table 3 LDR Source Images Used as Input of Implemented WDR Generation Methods

Name	Size (Height x Width)	Photographer	Bits per Color Channel	Exposure Time (s)
book	480 x 640	Author	8	1/20, 1/125, 1/640
lab1	576 x 704	Author	8	1/50, 1/100, 1/250
lab2	576 x 704	Hugo Edwar Benitez Leyva	8	1/50, 1/250, 1/700, 1/1000, 1/2500, 1/10000
window	683 x 1024	Mathiaz Ethiz	8	15, 4, 1, 1/4, 1/15, 1/60, 1/250, 1/1000, 1/4000

3.5 WDR Image Generation

3.5.1 MATLAB "makehdr"

```
files = {'office_1.jpg', 'office_2.jpg', 'office_3.jpg', ...
         'office_4.jpg', 'office_5.jpg', 'office_6.jpg'};

expTimes = [0.0333, 0.1000, 0.3333, 0.6250, 1.3000, 4.0000];

hdr = makehdr(files, 'RelativeExposure', expTimes ./ expTimes(1));
rgb = tonemap(hdr);
figure; imshow(rgb)
```

This function creates the single-precision WDR image from the set of LDR images. The source code can be referred at **Appendix A**. The function works as follows:

1. The inputs are parsed and checked.
2. The LDR images are passed through to find the lowest exposure image.
3. Output variables for an accumulator ("hdr") and the number of the LDR images that contributed to each pixel are created.
4. WDR image is constructed the by iterating over the LDR images.
    a. Over- and under-exposed values are removed.
    b. The intensity of the each LDR image is brought into a common domain by normalizing it using the relative exposure, and then it is added to the accumulator.
5. The values on the accumulator are averaged by the number of LDR images that contributed to each pixel to produce the WDR radiance map.
6. For pixels that were completely over-exposed, they are assigned with the maximum value computed for the properly exposed pixels.
7. For pixels that were completely under-exposed, they are assigned with the minimum value computed for the properly exposed pixels.
8. For pixels that were sometimes under-exposed, sometimes over-exposed, and never properly exposed, only their nonzero values will be considered.

9. A floating point accumulator for the final WDR image is created.

After floating point accumulator for final WDR image is created, WDR image is saved in form of *.hdr image file format using function of "hdrwrite". The source code can be referred at **Appendix I**.

3.5.2 P. Debevec and J. Malik Method

All equations discussed here are referred from [9]. The exposure X is the product of the irradiance, E at the film and exposure time, Δt, and its unit is Jm^{-2}.

$$X = E\Delta t \tag{1}$$

After digitization processes, digital number Z is obtained, which is a non-linear function, f of the original exposure X at the pixel. This function f will be determined and then exposure X at each pixel can be computed as follows

$$X = f^{-1}(Z) \tag{2}$$

It is assumed that the function f is monotonically increasing, thus its inverse, f^{-1} is well-defined. After determining exposure X and exposure time Δt, the irradiance E can be calculated as

$$E = \frac{X}{\Delta t} \tag{3}$$

where, E is assumed to be always proportional to the radiance L in the scene.

The input of the algorithm is a number of digitized images of a scene with different known exposure time Δt_j. The assumptions are:

i. The scene is static

ii. The process is done quickly enough that changes in lighting can be safety ignored.

iii. The irradiance values E_i for each pixel i are constant.

Z_{ij} is defined as pixel values where i is a spatial index over pixels and j is an index of exposure times Δt_j. The film reciprocity equation is

$$Z_{ij} = f(E_i \, \Delta t_j) \tag{4}$$

Since f is assumed to be monotonic and invertible, previous equation can be rewritten as

$$f^{-1}(Z_{ij}) = E_i \, \Delta t_j \tag{5}$$

$$\ln f^{-1}(Z_{ij}) = \ln E_i + \ln \Delta t_j \tag{6}$$

Then,

$$\ln f^{-1} = g \tag{7}$$

$$g(Z_{ij}) = \ln E_i + \ln \Delta t_j \tag{8}$$

where, i varies over pixels and j varies over exposure times. The known are pixel values Z_{ij} and exposure times Δt_j, and the unknown are irradiance E_i and function g. Next, $(Z_{max} - Z_{min} + 1)$ values of g(Z) and N values of In E_i are determined so that the following objective function is minimized.

The objective function is

$$O = \sum_{i=1}^{N} \sum_{j=1}^{P} [g(Z_{ij}) - \ln E_j - \ln \Delta t_j]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} g"(z)^2 \tag{9}$$

where, Z_{min} and Z_{max} are the lowest and highest pixel integer values, N is the number of pixel locations and P is the number of images. The first term checks in a least-squared sense if the solution satisfies the set equations derived from Equation 8. The second term is a smoothing function of g where

$$g"(z) = g(z-1) - 2g(z) + g(z+1) \tag{10}$$

The scalar λ weights the smoothing function relative to the data fitting term. For the amount of noise expected in Z_{ij}, λ should be chosen properly. Then, the singular value decomposition (SVD) method is used to solve for overdetermined system of linear equations. This well explanation of procedure is found in Figure 7 and Figure 8.

Solution of the g(z) and E_i values can be up to a single factor α. To set a scale factor, the additional constraint $g(Z_{mid}) = 0$ is added in the equation where

$$Z_{mid} = \frac{1}{2}(Z_{min} + Z_{max}) \tag{11}$$

A weighting function w(z) is then introduced to enhance the smoothness and emphasize the data fitting towards the middle of the curve. A simple hat function is chosen as a w(z).

$$w(z) = \begin{cases} z - Z_{min}, & z \le \frac{1}{2}(Z_{min} + Z_{max}) \\ Z_{max} - z, & x > \frac{1}{2}(Z_{min} + Z_{max}) \end{cases} \tag{12}$$

Then, the previous objective function O become

$$O = \sum_{i=1}^{N}\sum_{j=1}^{P}\{w(Z_{ij})\left[g(Z_{ij}) - \ln E_i - \ln \Delta t_j\right]\}^2 + \lambda \sum_{z=Z_{min+1}}^{Z_{max}-1}[w(z)g"(z)]^2 \tag{13}$$

From Equation 8,

$$\ln E_i = g(Z_{ij}) - \ln \Delta t_j \tag{14}$$

For effectiveness, w(z) is reused to emphasize the exposures in which the pixel values are closer to the middle of the response function. Finally, the radiance map is obtained in the form of

$$\ln E_i = \frac{\sum_{j=1}^{P} w(Z_{ij})[g(Z_{ij}) - \ln \Delta t_j]}{\sum_{j=1}^{P} w(Z_{ij})} \tag{15}$$

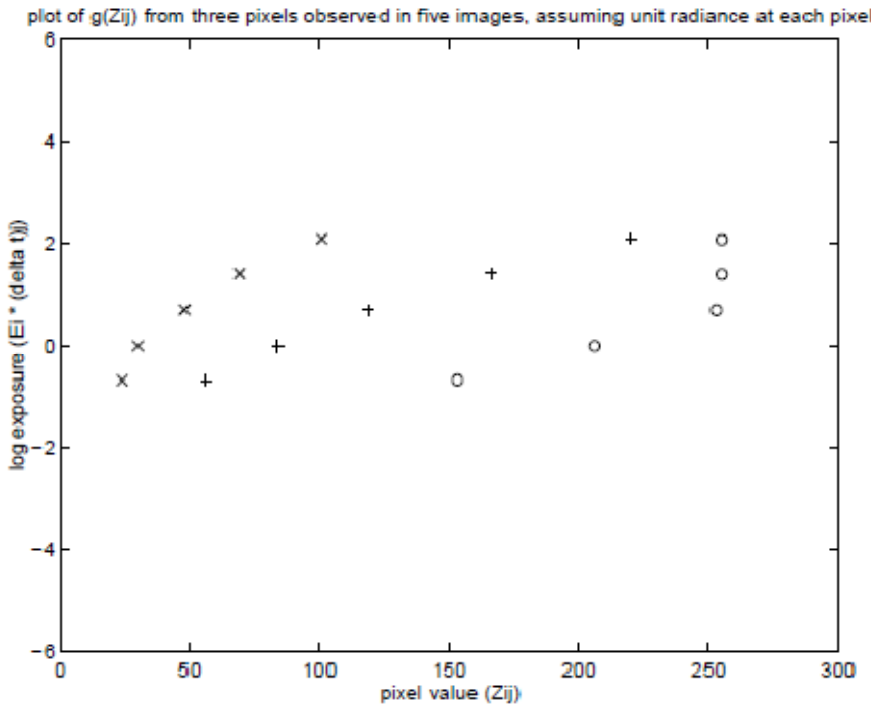


Figure 7 The sample of the g derived from values at three different pixels for five different known exposure using Equation 8 [9]

The x, + and o symbols represent samples of the g derived from values at 3 different pixels for 5 different known exposures using Equation 2. The unknown log irradiance ln E_i has been arbitrarily assumed to be 0. It is noted that the shape of g is correct, even though its vertical position is arbitrary corresponding to the unknown ln E_i. The 3 sample of the g curves segment are then shifted up and

down (by adjusting the ln E_i) until they line up into a single smooth, monotonic curve as shown in Figure 8.

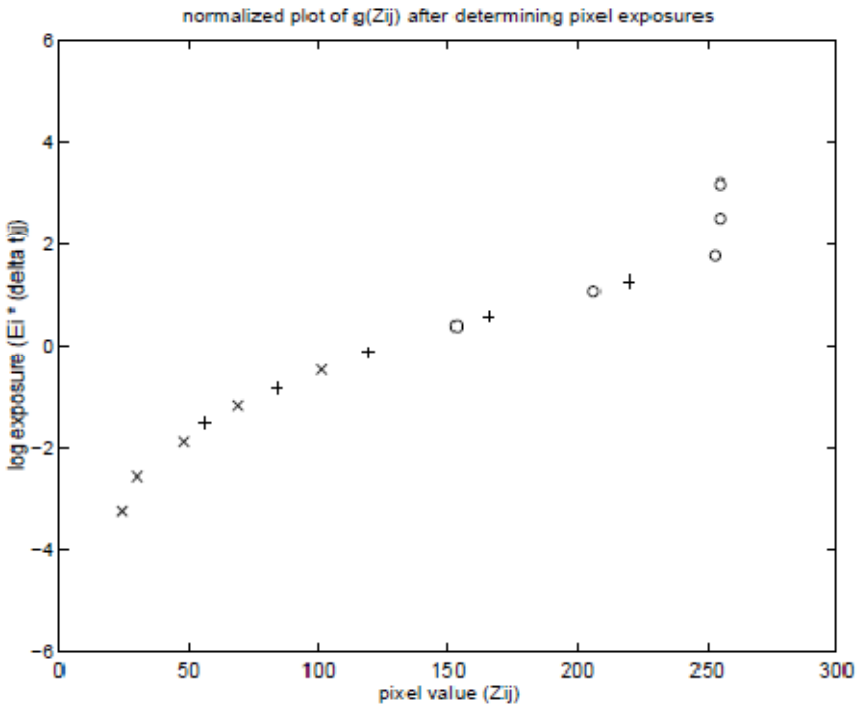


Figure 8 The optimized g curves produced from the line up sample of g curves of three pixels at different known exposures [9]

3.6 WDR Image

WDR image is the output resulted from each of previous WDR generation method. Then, all chosen WDR tone-mapping methods are applied to all of them. The involved WDR images are listed in the following Table 4. For bristolb.hdr [33], memorial.hdr and rosette.hdr [34], they are all downloaded from the website of famous WDRI researchers.

Table 4 WDR Images Used as Input of Implemented Tone-Mapping Algorithms

Name	Size (Height x Width)	Photographer	Generation Method	Bits per Color Channel
book	480 x 640	Author	All methods which are chosen	32
bristolb	1536 x 2048	Greg Ward	N/A	32
lab1	576 x 704	Author	All methods which are chosen	32
lab2	576 x 704	Hugo Edwar Benitez Leyva	All methods which are chosen	32
memorial	768 x 512	Paul Debevec	N/A	32
rosette	480 x 720	Paul Debevec	N/A	32
window	683 x 1024	Mathiaz Ethiz	All methods which are chosen	32

3.7 WDR Visualization (Tone-Mapping)

3.7.1 Logarithmic

All equations discussed here are referred from [6]. The radiance map of WDR image is converted into luminance map as follows

$$L_w(x,y) = 0.270R + 0.670G + 0.060B$$

Range compression is done by mapping the luminance as follows

$$L_d(x,y) = \frac{log\ _{10}\big(1 + L_w(x,y)\big)}{log_{10}(1 + L_{max})}$$

Next, the luminance channels are recombined by using the following per-channel gamma correction

$$Compressed\ Red\ Channel = L_d\left(\frac{R_w}{L_w}\right)^s$$

$$Compressed\ Green\ Channel = L_d\left(\frac{G_w}{L_w}\right)^s$$

$$Compressed\ Blue\ Channel = L_d\left(\frac{B_w}{L_w}\right)^s$$

3.7.2 Modified Logarithmic

The radiance map of WDR image is converted into luminance map as follows, but this time, each color channel of the luminance map is treated differently for experimental and comparison purposes.

$$L_{w_R}(x,y) = 0.270R$$

$$L_{w_G}(x,y) = 0.670G$$

$$L_{w_B}(x,y) = 0.060B$$

Range compression is done by mapping the luminance as follows

$$L_{d_R}(x,y) = \frac{log\ _{10}\left(1 + L_{w_R}(x,y)\right)}{log_{10}\left(1 + L_{max_R}\right)}$$

$$L_{d_G}(x,y) = \frac{log\ _{10}\left(1 + L_{w_G}(x,y)\right)}{log_{10}\left(1 + L_{max_G}\right)}$$

$$L_{d_B}(x,y) = \frac{log\ _{10}\left(1 + L_{w_B}(x,y)\right)}{log_{10}\left(1 + L_{max_B}\right)}$$

Next, the luminance channels are recombined by using the following per-channel gamma correction

$$Compressed\ Red\ Channel = L_{d_R}\left(\frac{R_w}{L_{w_R}}\right)^s$$

$$Compressed\ Green\ Channel = L_{d_G}\left(\frac{G_w}{L_{w_G}}\right)^s$$

$$Compressed\ Blue\ Channel = L_{d_B}\left(\frac{B_w}{L_{w_B}}\right)^s$$

3.7.3 Exponential Tone-Mapping

All equations discussed here are referred from [6]. The radiance map of WDR image is converted into luminance map as follows

$$L_w(x,y) = 0.270R + 0.670G + 0.060B$$

Range compression is done by mapping the luminance as follows

$$L_d(x,y) = 1 - exp\left(-\frac{L_w(x,y)}{L_{av}}\right)$$

where, L_{av} is average luminance and δ is small value (0.0001) to avoid equation goes to infinity if black pixels (zero) are present in the image:

$$L_{av} = \left(\frac{1}{N}\sum_{x,y} log\big(\delta + L_w(x,y)\big)\right)$$

Next, the luminance channels are recombined by using the following per-channel gamma correction

$$Compressed\ Red\ Channel = L_d\left(\frac{R_w}{L_w}\right)^s$$

$$Compressed\ Green\ Channel = L_d\left(\frac{G_w}{L_w}\right)^s$$

$$Compressed\ Blue\ Channel = L_d\left(\frac{B_w}{L_w}\right)^s$$

For experimental purpose, value of δ will be set to be 0.001, 0.01 and 0.1 to see its effects on output images.

3.7.4 Modified Exponential

The radiance map of HDR image is converted into luminance map as follows, but this time, as applied to the previous Logarithmic tone-mapping, each color channel of the luminance map is treated differently for experimental and comparison purposes.

$$L_{w_R}(x,y) = 0.270R$$

$$L_{w_G}(x,y) = 0.670G$$

$$L_{w_B}(x,y) = 0.060B$$

Range compression is done by mapping the luminance as follows

$$L_{d_R}(x,y) = 1 - exp\left(-\frac{L_{w_R}(x,y)}{L_{av_R}}\right)$$

$$L_{d_G}(x,y) = 1 - exp\left(-\frac{L_{w_G}(x,y)}{L_{av_G}}\right)$$

$$L_{d_B}(x,y) = 1 - exp\left(-\frac{L_{w_B}(x,y)}{L_{av_B}}\right)$$

where, L_{av_R}, L_{av_G} and L_{av_B} are average luminance for each color channel and δ is small value to avoid equation goes to infinity if black pixels are present in the image:

$$L_{av_R} = \left(\frac{1}{N}\sum_{x,y} log\left(\delta + L_{w_R}(x,y)\right)\right)$$

$$L_{av_G} = \left(\frac{1}{N}\sum_{x,y} log\left(\delta + L_{w_G}(x,y)\right)\right)$$

$$L_{av_B} = \left(\frac{1}{N}\sum_{x,y} log\left(\delta + L_{w_B}(x,y)\right)\right)$$

Next, the luminance channels are recombined by using the following per-channel gamma correction

$$Compressed\ Red\ Channel = L_{d_R}\left(\frac{R_w}{L_{w_R}}\right)^s$$

$$Compressed\ Green\ Channel = L_{d_G}\left(\frac{G_w}{L_{w_G}}\right)^s$$

$$Compressed\ Blue\ Channel = L_{d_B}\left(\frac{B_w}{L_{w_B}}\right)^s$$

Also for experimental purpose, value of δ will set to be equal to 0.001, 0.01 and 0.1 to see its effects on output images.

3.7.5 Reinhard et al Global Tone-Mapping Operator

All equations discussed here are referred from [4]. The radiance map of WDR image is converted into luminance map as follows

$$L_w(x,y) = 0.270R + 0.670G + 0.060B$$

Initial scaling of global operator:

$$L_m(x,y) = \frac{a}{L_{av}} L_w(x,y)$$

where,

$$L_{av} = \left(\frac{1}{N} \sum_{x,y} log\big(\delta + L_w(x,y)\big) \right)$$

To predominantly compress the high luminance regions near highlights or in the sky

$$L_d(x,y) = \frac{L_m(x,y)}{1 + L_m(x,y)}$$

Now, the luminance values are mapped to display range between 0 and 1. But in practice, largest display luminance values do not quite reach 1. In the following equation, L_{white} allows this global operator to control the smallest luminance value that will be mapped to white. By default, L_{white} is set to the maximum of scaled world luminance, $L_{m,max}$.

$$L_d(x,y) = \frac{L_m(x,y)\left(1 + \frac{L_m(x,y)}{L_{white}^2}\right)}{1 + L_m(x,y)}$$

3.7.6 Reinhard et al Local Tone-Mapping Operator

All equations discussed here are referred from [4]. If the Gaussion-blurred image at scale s is given by

$$L_s^{blur}(x,y) = L_m(x,y) \otimes R_s(x,y)$$

the center-surround value at that scale is calculated as

$$V_s(x,y) = \frac{L_s^{blur} - L_{s+1}^{blur}}{2^{\phi}\, a/s^2 + L_s^{blur}}$$

To find the largest area that has relatively low contrast, we search for largest scale s_{max} for which the Gaussians difference is always below a threshold ϵ,

$$s_{max}: \left|V_{s_{max}}(x,y)\right| < \epsilon$$

Finally, local operator

$$L_d(x,y) = \frac{L_m(x,y)}{1 + L_{s_{max}}^{blur}(x,y)}$$

3.7.7 Garrett et al Method

All equations discussed here are referred from [28]. The input of this algorithm is an RGB HDR image. Firstly, it is converted into CIE 1931 tristimulus values.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 41.384 & 22.155 & 0.487 \\ 25.053 & 51.424 & 5.438 \\ 11.014 & 9.743 & 56.089 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The chromatic adaptation transform

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = M_{CAT02} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

where,

$$M_{CAT02} = \begin{bmatrix} 0.7328 & 0.4296 & -0.1624 \\ -0.7036 & 1.6975 & 0.0061 \\ 0.0030 & 0.0136 & 0.9834 \end{bmatrix}$$

The linear von Kries transform with incomplete adaptation term D

$$R_c = \left[\left(Y_w \frac{D}{R_w}\right) + (1 - D)\right] R$$

$$G_c = \left[\left(Y_w \frac{D}{G_w}\right) + (1 - D)\right] G$$

$$B_c = \left[\left(Y_w \frac{D}{B_w}\right) + (1 - D)\right] B$$

where, the whitepoint for D65 is

$$Y_w = [95.05 \quad 100.00 \quad 108.88]$$

and whitepoints of the image are obtained from localized adaptation (luminance only adaptation) that uses a low-pass version of the image itself as adapting whitepoint.

$$\begin{bmatrix} R_w \\ G_w \\ B_w \end{bmatrix} = M_{CAT02} \cdot \begin{bmatrix} Y \\ Y \\ Y \end{bmatrix}_{filtered}$$

$$\begin{bmatrix} Y \\ Y \\ Y \end{bmatrix}_{filtered} = FFT^{-1}\left(FFT \begin{bmatrix} Y \\ Y \\ Y \end{bmatrix} FFT\left(e^{\left(-\frac{x+y}{\sigma}\right)^2} \right) \right)$$

Then, $R_c G_c B_c$ velues are converted into XYZ tristimulus values.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = {M_{CAT02}}^{-1} \begin{bmatrix} R_c \\ G_c \\ B_c \end{bmatrix}$$

Next, XYZ tristimulus values are converted into LMS cone responses to prepared it for local contrast adjustments.

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.4002 & 0.7077 & -0.0807 \\ -0.2280 & 1.1500 & 0.0612 \\ 0.0000 & 0.0000 & 0.9184 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

The local contrast adjustment is described as follows

$$F_L = 0.2\left(\frac{1}{5L_A}\right)^2 (5L_A) + 0.1\left(1 - \left(\frac{1}{5L_A + 1}\right)^4\right)^2 (5L_A)^{\frac{1}{3}}$$

$$F_{L,scaled} = (1/1.7)F_L$$

where,

$$L_A = \begin{bmatrix} Y \\ Y \\ Y \end{bmatrix}_{filtered}$$

Then, the LMS cone responses are compressed.

$$\begin{bmatrix} L' \\ M' \\ S' \end{bmatrix} = \begin{bmatrix} L^{0.43 \cdot F_L} \\ M^{0.43 \cdot F_L} \\ S^{0.43 \cdot F_L} \end{bmatrix}$$

The LMS cone responses are converted into IPT appearance space to calculate the appearance correlates.

$$\begin{bmatrix} I \\ P \\ T \end{bmatrix} = \begin{bmatrix} 0.4000 & 0.4000 & 0.2000 \\ 4.550 & -4.8510 & 0.3960 \\ 0.8056 & 0.3572 & -1.1628 \end{bmatrix} \cdot \begin{bmatrix} L' \\ M' \\ S' \end{bmatrix}$$

Then, the IPT appearance space is converted back into LMS cone responses.

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.4000 & 0.4000 & 0.2000 \\ 4.550 & -4.8510 & 0.3960 \\ 0.8056 & 0.3572 & -1.1628 \end{bmatrix}^{-1} \cdot \begin{bmatrix} I \\ P \\ T \end{bmatrix}$$

LMS cone responses are converted back into XYZ tristimulus values.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4002 & 0.7076 & -0.0808 \\ -0.2280 & 1.1500 & 0.0612 \\ 0.0000 & 0.0000 & 0.9184 \end{bmatrix}^{-1} \cdot \begin{bmatrix} L^{\frac{1}{0.43}} \\ M^{\frac{1}{0.43}} \\ S^{\frac{1}{0.43}} \end{bmatrix}$$

XYZ tristimulus values are converted back into RGB color space.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 41.384 & 22.155 & 0.487 \\ 25.053 & 51.424 & 5.438 \\ 11.014 & 9.743 & 56.089 \end{bmatrix}^{-1} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

For better result, extreme bright pixels must be removed by using the following clipping function.

$$Clip: RGB < 99\% = RGB$$

$$RGB > 99\% = 99\%$$

Finally, RGB values are scaled into ranges between 0 and 1 and compressed using a power function and scaled into ranges between 0 and 255.

$$RGB = \frac{RGB - RGB_{min}}{RGB_{max} - RGB_{min}}$$

$$dRGB = 255 \cdot RGB^{\frac{1}{1.7}}$$

3.7.8 MATLAB "tonemap"

```
hdr = hdrread('office.hdr');
imshow(hdr)
rgb = tonemap(hdr, 'AdjustLightness', [0.1 1], ...
                   'AdjustSaturation', 1.5);
figure;
imshow(rgb)
```

This function converts WDR image to LDR image. The source code can be referred at **Appendix J**. The function works as follows:

1. Input arguments are parsed and validated.
2. WDR image is transformed to be in the range [0,1] by taking the base-2 logarithm and linearly scaling it.
    a. 0's from each channel is removed. This unlikely to have a significant effect on color quality and prevents log of 0 goes to infinity.
    b. The log-luminance histogram is normalized to [0,1] using Ward's simple method.
3. The WDR image is converted to LDR image using adaptive histogram equalization. The image is always assigned to be in the range [0,1].
    a. The log-luminance image is converted to XYZ image by assuming D65 whitepoint.
    b. The XYZ image is converted to L*a*b* image using D65 whitepoint.
    c. The L* values is tone-mapped to preserve overall color as much as possible.
    d. This decreases global saturation. Therefore, a* and b* channels are scaled so that global saturation is reintroduced.
    e. The L*a*b* image is converted back to XYZ using D65 whitepoint.
    f. The XYZ image is converted back to sRGB using D65 whitepoint.
4. sRGB image is scaled to be in the range [0,255]. LDR image is obtained.

3.8 LDR Image

The resulted output images are 8 bits per color channel which means that it can be displayed directly and properly on the LDR display device.

3.9 Analysis

3.9.1 Objective Measure (Computation Time)

Computation time for each method or algorithm is recorded using MATLAB "tic" and "toc" timer function.

```
% Start timer
tic;

% Algorithm
...

% End timer
time = toc;
```

3.9.2 Subjective Measure (Observer Evaluation)

The output images were evaluated by twelve persons. The rating scale used is based on "Rating scale of the Television Allocations Study Organization (Frendendall and Brehrend)" [15]. It is as shown in Table 5 below. Evaluation results were then averaged and standard deviations were calculated.

Table 5 Rating Scale Used in Observer Evaluation

Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Output Images (LDR Images)

4.1.1 Set 1: Standard WDR Images Tone-Mapping

For Set 1, standard WDR images generated by famous WDRI researchers [33, 34] were tone-mapped using the selected tone-mapping methods. They are "bristolb.hdr", "memorial.hdr" and "rosette.hdr". All output images are tabulated in **Appendix FF** until **Appendix HH**. They are sorted by image so that the differences resulted from all tone-mapping methods can be seen.

Logarithmic tone-mapping produces images which are dark and tend to let domination of dark pixels. Thus, very little details are preserved.

Modified logarithmic tone-mapping behaves similar to logarithmic tone-mapping. The produced images are dark and have very minimum amount of exposed details. It is also observed that modified logarithmic tone-mapping produces bluish images.

Exponential tone-mapping gives good results even though there are some losses of image details in bright and dark regions. However, the losses are acceptable to a certain limit. Increasing value of δ gives darker images.

Modified tone-mapping performs worse than exponential tone-mapping because more image details lost in bright regions for same value of δ. Like exponential tone-mapping, increasing value of δ gives darker images. On the other hand, color appearance become less natural for smaller value of δ.

Reinhard et al global tone-mapping produces similar results as exponential tone-mapping ($\delta = 0.0001$) does but with fewer losses of image details and color of images produced are more saturated than others.

If compared to Reinhard et al global tone-mapping, Reinhard et al local tone-mapping produces images with sharper details and less saturated color. It reveals more details in both bright and dark regions.

Garrett et al tone-mapping produces images with grayish and a bit yellowish color and does not working well when dealing with preservation of image details. Also, bright regions are prone to have dark regions around it.

MATLAB function reveals most details in both bright and dark regions. Even so, it produces images that have less saturated or pale color. This is obviously seen at "rosette".

Photomatix details enhancer preserves image details sufficiently. On the other hand, there is a noticeable trade-off between details preservation and perceived brightness. The output images have to become a bit less bright in order to reveal the image details. This trade-off can be seen at "memorial" and "rosette".

Photomatix tone compressor has produced output images at satisfactory level or details preservation. Even so, the output images produced is less bright than Photomatix details enhancer. This is obvious for "bristolb" where details in dark regions are under-exposed.

4.1.2 Set 2: WDR Image Generation Using MATLAB "makehdr" Function Followed by Tone-Mapping Methods

For Set 2, WDR images were generated using MATLAB "makehdr" function followed by the selected tone-mapping methods. They are "book.hdr", "lab1.hdr", "lab2.hdr" and "window.hdr". All output images are tabulated in **Appendix II** until **Appendix LL**. They are sorted by image so that the differences resulted from all tone-mapping methods can be seen.

Logarithmic tone-mapping produces low contrast images for "book", "lab1", and "lab2". However, image details still can be seen. On the other hand,

for “window”, logarithmic dark image is produced. Thus, there are undesirable losses of details in dark regions.

Modified logarithmic tone-mapping behaves similar to logarithmic tone-mapping. For “book”, “lab1” and “lab2”, produced images are low contrast and losses of image details are obvious. For “window”, dark image is produced and losses of details are very high. It is observed that modified logarithmic tone-mapping produces pinkish images.

Exponential tone-mapping produces images with minimal losses of details. Varying δ value doesn’t really affect the brightness of the produced images for “book”, “lab1” and “lab2”. However, for “window”, increasing value of δ produces darker image.

Modified exponential tone-mapping performs worse than exponential tone-mapping since more image details are lost in bright regions and scenes’ outdoor for same value of δ. Like exponential tone-mapping, varying δ value doesn’t affect the brightness of the produced images for “book”, “lab1”, and “lab2”. However, for “window”, increasing value of δ produces darker image. In addition, color appearance of the produced images doesn’t match the original scene. “book”, “lab1” and “lab2” are purplish while “window” appears with more odd color as δ value decreases.

Reinhard et al global tone-mapping produces similar results as exponential tone-mapping ($\delta = 0.0001$) does but with fewer losses of image details and more saturated color.

If compared to Reinhard et al global tone-mapping, Reinhard et al local tone-mapping produces images with sharper details and less saturated color. It reveals more details in both bright and dark regions.

Garrett et al tone-mapping produces images with grayish and slightly yellowish images and does not working well when dealing with preservation of image details. Also, as shown by “book” and “window”, bright regions are prone to have dark regions around it.

If compared to Reinhard et al global tone-mapping, MATLAB function produces images with similar level of details preservation but with less saturated color.

Photomatix details enhancer produces images with similar level of details preservation as Reinhard et al local tone-mapping but the images are brighter. However, as seen at "window", there is a noticeable trade-off between details preservation and perceived brightness. The output image has to become less bright in order to reveal the image details.

For "book", "lab1" and "lab2", Photomatix tone compressor has produced output images which are similar to Reinhard et al local tone-mapping but with less sharper details. For "window", bright image is produced with bit losses of details in bright regions.

For Set 2, posterization (abrupt changes from one to another) can be seen in all "window" images except those that resulted from logarithmic and modified.

4.1.3 Set 3: WDR Image Generation Using P. Debevec and J. Malik Method Followed by Tone-Mapping Methods

For Set 3, WDR images were generated using method of P. Debevec and J. Malik followed by the selected tone-mapping methods. They are "book.hdr", "lab1.hdr", "lab2.hdr" and "window.hdr". All output images are tabulated in **Appendix MM** until **Appendix PP**. They are sorted by image so that the differences resulted from all tone-mapping methods can be seen.

For "book", logarithmic tone-mapping produces low contrast image but preservation of details is quite good. For "lab1", "lab2" and "window", this tone-mapping method tends to let domination of dark regions. Thus, little details are preserved.

Modified logarithmic tone-mapping behaves similar to logarithmic tone-mapping. The produced images are low contrast for "book" but dark for "lab1", "lab2" and "window". They have little amount of exposed details. It is observed that modified logarithmic tone-mapping produces pinkish images.

Exponential tone-mapping gives good results for “book” and “window” even though there are some image details lost in bright regions. For “lab1” and “lab2”, more image details lost in both bright and dark regions. In addition, their color appearance is not preserved. Apart from that, varying δ value doesn’t affect the brightness of all produced images.

Modified exponential tone-mapping performs worse than exponential tone-mapping do because more image details are lost in bright regions for same value of δ. Like exponential tone-mapping, varying δ value doesn’t really affect the brightness of the produced images. In addition, color appearance of the produced images doesn’t match the original scene since “book”, “lab1” and “lab2” are pinkish while “window” appear with odd color.

Reinhard et al global tone-mapping produces similar results as exponential tone-mapping does but with fewer losses of image details in bright regions. For “book” and “window”, the color is more saturated than others. Like exponential tone-mapping, color appearance is also not preserved for “lab1” and “lab2”.

If compared to Reinhard et al global tone-mapping, Reinhard et al local tone-mapping produces sharper images but less saturated in color. However, like Reinhard et al global tone-mapping, color appearance is not preserved for “lab1” and “lab2”.

Garrett et al tone-mapping produces images with grayish and slightly yellowish images and does not working well when dealing with preservation of image details. As seen at “book” and “window”, losses of details are acceptable but bright regions are prone to have dark regions around it. For “lab1” and “lab2”, the images are darker and there losses of details in dark regions.

MATLAB function preserves details sufficiently but if compare to others the color is less saturated.

Like MATLAB function, Photomatix details enhancer has preserved details sufficiently but with some acceptable losses in dark regions.

If compared to Photomatix details enhancer, Photomatix tone compressor gives slightly darker image with more losses in dark regions.

4.1.4 Set 4: WDR Image Generation Using Photomatix WDR Generator Followed by Tone-Mapping Methods

For Set 4, WDR images were generated using Photomatix WDR image generator followed by the selected tone-mapping methods. They are "book.hdr", "lab1.hdr", "lab2.hdr" and "window.hdr". All output images are tabulated in **Appendix QQ** until **Appendix TT**. They are sorted by image so that the differences resulted from all tone-mapping methods can be seen.

Logarithmic tone-mapping produces dark images for "book", "lab1" and "lab2" and even darker image for "window". As seen at all output images, a lot details lost in dark region and very little image details are preserved.

Modified logarithmic tone-mapping behaves similar to logarithmic tone-mapping. The produced images are dark for "book", "lab1" and "lab2" and even darker for "window". All of them have very little amount of well-exposed details. It is observed that modified logarithmic tone-mapping produces purplish images.

Exponential tone-mapping gives good results for "book" and "window" even though there are some image details lost in bright regions. For all images, increasing δ value increases the visibility of details in bright regions and scenes' outdoor while decreasing δ value increases the visibility of details of in dark regions and scenes' indoor.

Modified exponential tone-mapping performs slightly worse than exponential tone-mapping because more image details are lost in bright regions for same value of δ. In addition, color appearance of the produced images doesn't match the original scene. For "book", "lab1" and "lab2", increasing δ value produces darker images but with more natural color while decreasing δ value produces more purplish images. For "window", increasing δ value also produces darker image but with more natural color while decreasing δ value produces image with more odd color. Like exponential tone-mapping, for all images, increasing δ value increases the visibility of details in bright regions and scenes'

outdoor while decreasing δ value increases the visibility of details of in dark regions and scenes' indoor.

Reinhard et al global tone-mapping produces similar results as exponential tone-mapping (δ = 0.0001) does but with fewer losses of image details and more saturated color.

If compared to Reinhard et al global tone-mapping, Reinhard et al local tone-mapping produces sharper images that are less saturated in color. It reveals more details in both bright and dark regions.

Garrett et al tone-mapping produces images with grayish and a bit yellowish dull images and does not working well when dealing with preservation of image details. As seen at "book" and "window", losses of details are acceptable but bright regions are prone to have dark regions around it. For "lab1" and "lab2", the images are darker and there losses of details in dark regions.

In term of details preservation, function behaves similarly like Reinhard et al global tone-mapping. It preserves details sufficiently but if compare to others the color is less saturated.

For "book", "lab1" and "lab2", Photomatix details enhancer produces images with similar level of details preservation as Reinhard et al local tone-mapping but the images are brighter. However, as seen at "window", there is a noticeable trade-off between details preservation and perceived brightness. The output image has to become less bright in order to reveal the image details. For all images, they are less sharper than those produced by Reinhard et al details enhancer.

If compared to Photomatix details enhancer, Photomatix tone compressor gives slightly darker image with more losses in dark regions.

4.2 Objective Measure (Computation Time)

Computation time is recorded to indentify how complex an algorithm is. The computer used in order to complete the calculations is powered by Intel® Core™ i7 Processor i7-860 processor and has 6144 MB DDR3 RAM.

4.2.1 WDR Image Generation

Appendix UU shows the recorded computation times for all implemented WDR image generation algorithms. Also, Table 6 shows the average computation time for all implemented tone-mapping methods.

Table 6 Average Computation Time of All Implemented WDR Image Generation Methods

WDR Image Generation Method	Average Computation Time (seconds)			
	book	lab1	lab2	window
MATLAB "makehdr"	0.2206	0.2437	0.3980	1.0971
P. Debevec and J. Malik	1.7767	1.7637	1.0228	2.1434

For the same input image, when comparing between computation times of different algorithms implemented, it is observed that the most complex algorithm which is method of P. Debevec and J. Malik, takes the longest time to complete its computation.

For the same algorithm, when comparing between computation times for different input images used, it is observed that "window" which has largest size consumes the longest time to complete its WDR image generation process.

Conclusively, the computation time is affected by the algorithm complexity and the size of input images used.

4.2.2 WDR Image Visualization (Tone-Mapping)

Appendix VV shows the recorded computation times for all implemented tone-mapping algorithms. Also, Table 7 shows the average computation time for all implemented tone-mapping methods.

Table 7 Average Computation Time of All Implemented Tone-Mapping

Methods

Tone-Mapping Method	Average Computation Time (seconds)						
	bristolb	memorial	rosette	book	lab1	lab2	window
Logarithmic	1.5629	0.2937	0.2361	0.2978	0.4963	0.4568	0.8240
Modified Logarithmic	2.0474	0.3515	0.2911	0.3478	0.5305	0.5073	0.9066
Exponential	1.5076	0.2814	0.2155	0.3002	0.4896	0.4468	0.8077
Modified Exponential	1.9678	0.3433	0.2894	0.3466	0.5263	0.4962	0.8868
Reinhard et al Global Operator	1.5299	0.2889	0.2358	0.3006	0.4826	0.4529	0.8024
Reinhard et al Local Operator	55.8975	6.1355	5.5046	4.8937	6.5982	6.4099	12.4011
Garrett et al	7.4292	1.2812	0.9536	0.9319	1.1015	1.5451	2.2803
MATLAB “tonemap”	5.6236	0.8362	0.7900	0.7341	1.1178	0.9680	1.7855

For the same input image, when comparing between computation times of different algorithms implemented, it is observed that the most complex algorithm

which is Reinhard et al local tone-mapping operator, takes the longest time to complete its computation.

For the same algorithm, when comparing between computation times for different input images used, it is observed that bristolb.hdr which has largest size consumes the longest time to complete it tone-mapping process.

Conclusively, the computation time is affected by the algorithm complexity and the size of input images used.

4.3 Subjective Measure (Observer Evaluation)

All output images correspond to each algorithm are then evaluated by twelve persons using a rating scale stated in Section 3.9.2. The purpose of this evaluation is to rank the implemented algorithms according to how observers rate the resulted output images. **Appendix WW** and Figure 9 until Figure 26 show the evaluation results of all sets of implementation and they are grouped according to the input images used.

4.3.1 Set 1: Tone-Mapping Standard WDR Images

For "bristolb", Photomatix details enhancer shows the best result followed by Reinhard et al local operator and Reinhard et al global operator. They have produced output images which have maximum exposed details and color preservation. (Please refer to **Appendix WW**)

On the other hand, the worst results are shown by logarithmic, modified exponential (δ=0.0001), modified exponential (δ=0.01), modified exponential (δ=0.001) and Garrett et al tone-mapping. Logarithmic and Garret et al tone-mapping have produced dark and under-exposed images while another three tone-mapping methods have produced washed-out images with less color preservation. Figure 9 shows graphical representation of the computated average rating.

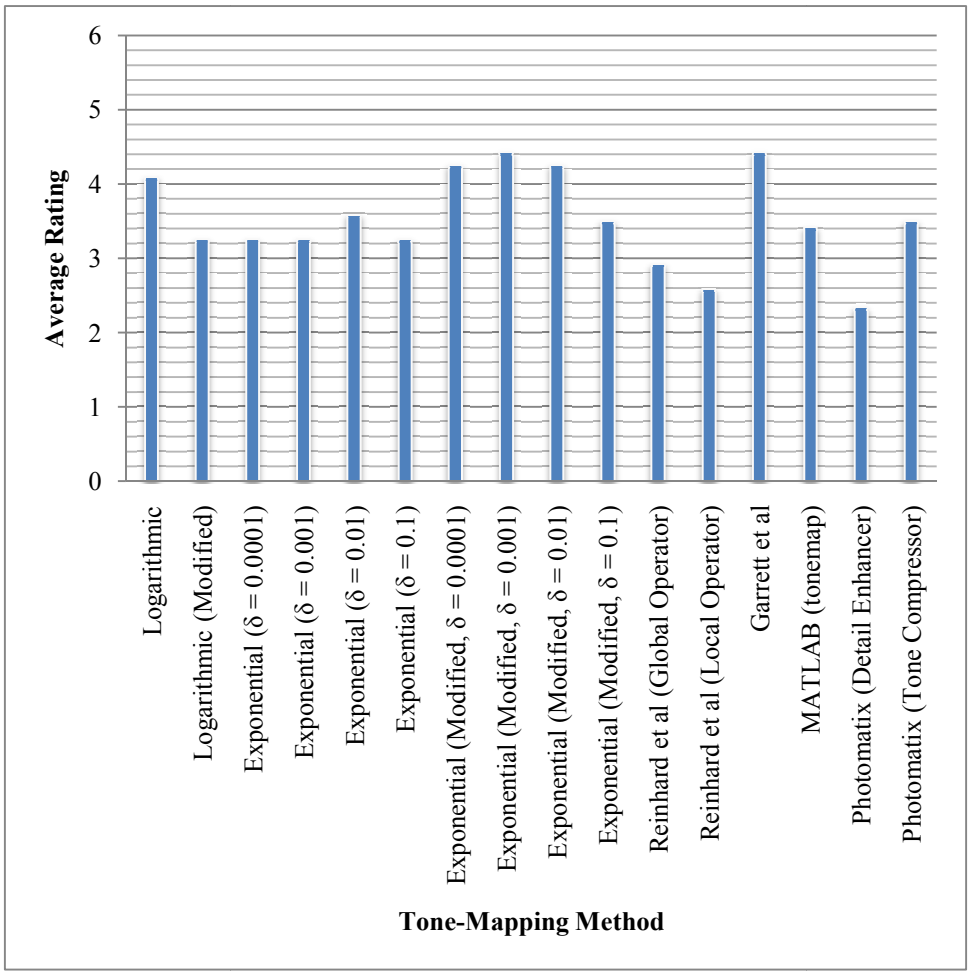


Figure 9 Average Rating of Set 1 (bristolb)

For "memorial", Photomatix tone compressor shows the best result followed by Reinhard et al global operator and exponential (δ=0.01). They have produced output images which have maximum exposed details and color preservation. (Please refer to **Appendix WW**)

The worst results are shown by modified exponential (δ=0.0001), logarithmic, and modified logarithmic tone-mapping. Modified exponential (δ=0.0001) has produced the most washed-out image which has maximum loss of image details in bright regions. Also, the color of image produced is a bit odd. Then, for logarithmic and modified logarithmic tone-mapping, they have produced dark and under-exposed images. Figure 10 shows graphical representation of the computated average rating.

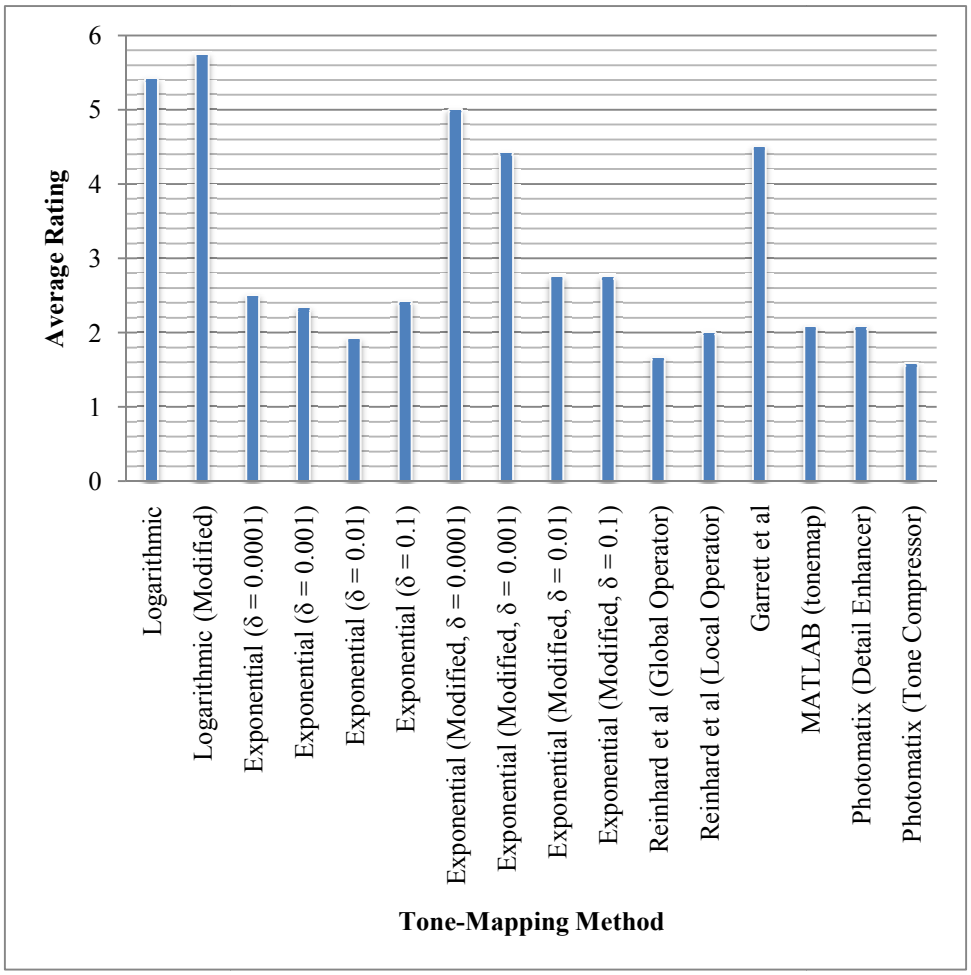


Figure 10 Average Rating of Set 1 (memorial)

For "rosette", Reinhard et al global operator shows the best result followed by exponential (δ=0.01) and Photomatix details enhancer. They have produced output images which have maximum exposed details and color preservation. (Please refer to **Appendix WW**)

The worst results are shown by modified logarithmic, modified exponential (δ=0.0001), logarithmic, and MATLAB tone-mapping. Logarithmic and modified logarithmic tone-mapping have produced dark and under-exposed images. Again, modified exponential (δ=0.0001) has produced the most washed-out image which has maximum loss of image details in bright regions and the color produced is a bit odd. For MATLAB tone-mapping, it results a dull image which has no preservation of color. However, loss of image details is minimum. Figure 11 shows graphical representation of the computated average rating.

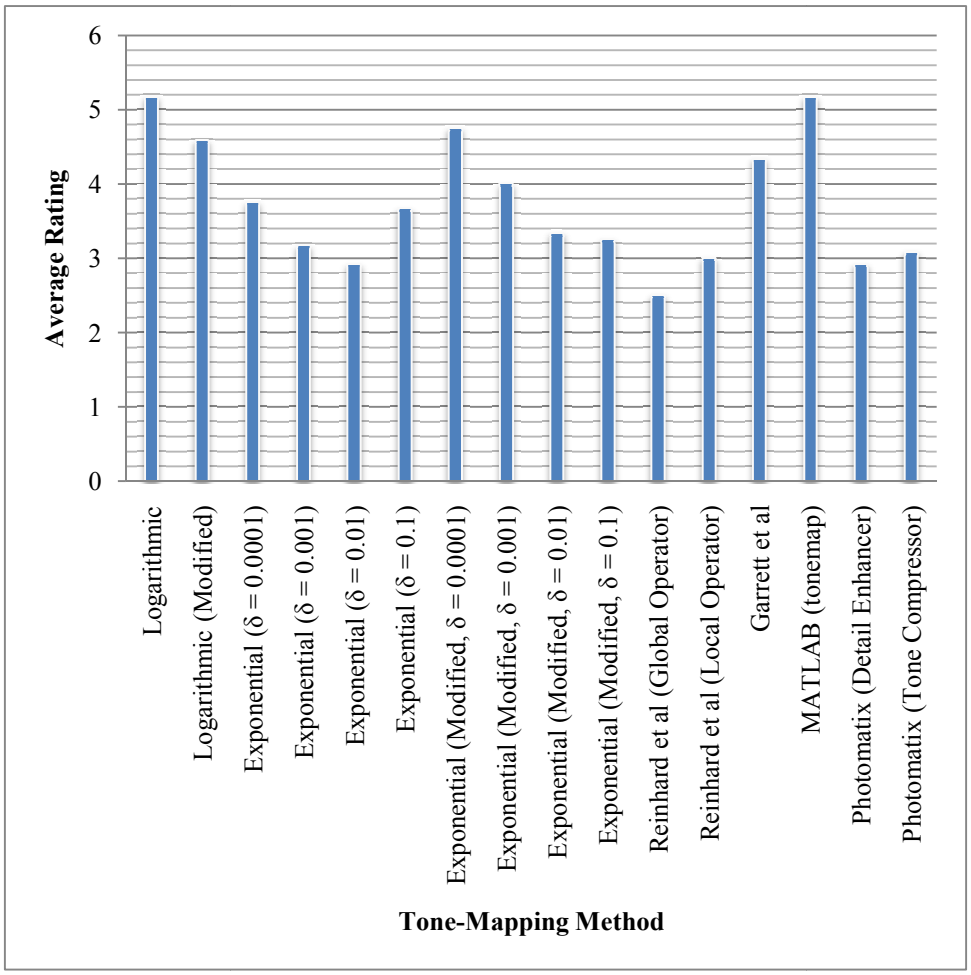


Figure 11 Average Rating of Set 1 (rosette)

Figure 12 shows graphical representation of the computated average rating for all involved images.

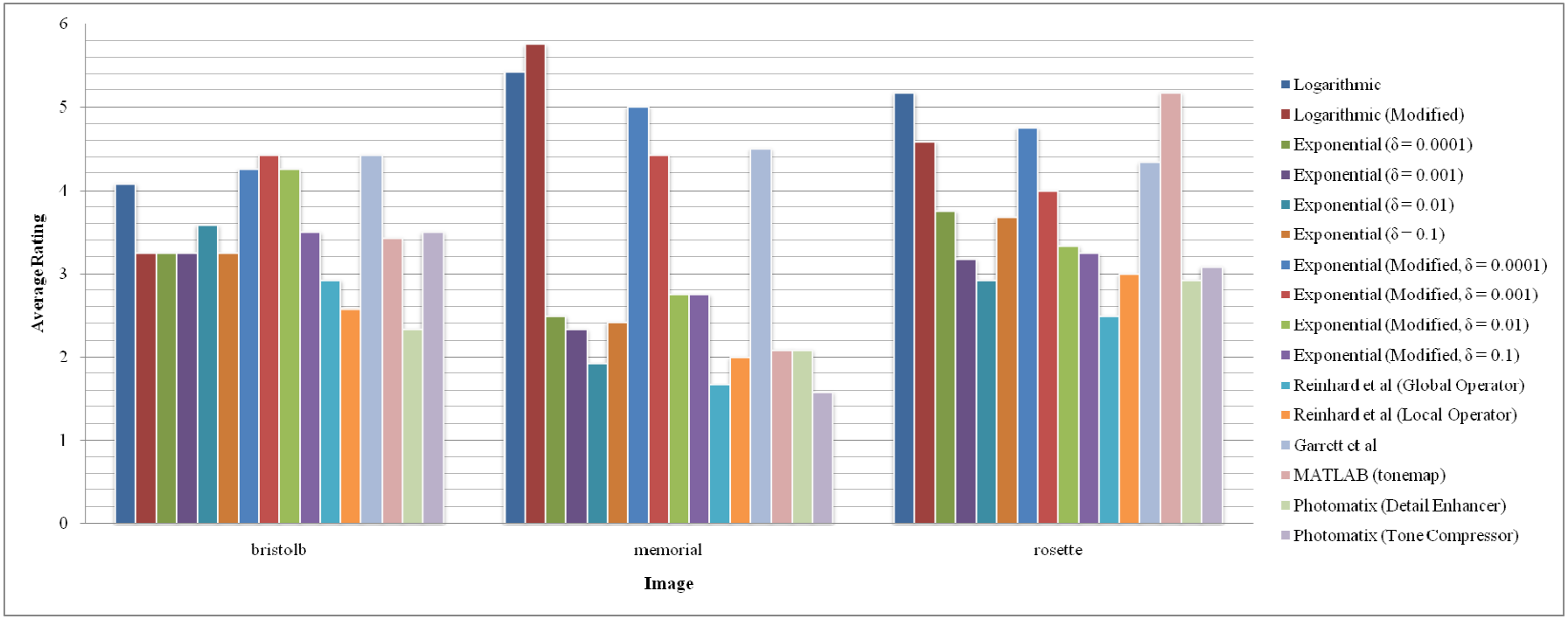


Figure 12 Average Rating of Set 1 (All)

In Table 8, overall average rating are calculated and in Table 9, the algorithms are ranked according their overall average rating. Recall that, smaller rating values means higher quality of images.

Table 8 Overall Average Rating for Set 1

Method	Average Rating			Overall Average Rating
	bristolb	memorial	rosette	
Logarithmic	4.08333	5.41667	5.16667	4.88889
Modified Logarithmic	3.25000	5.75000	4.58333	4.52778
Exponential ($\delta = 0.0001$)	3.25000	2.50000	3.75000	3.16667
Exponential ($\delta = 0.001$)	3.25000	2.33333	3.16667	2.91667
Exponential ($\delta = 0.01$)	3.58333	1.91667	2.91667	2.80556
Exponential ($\delta = 0.1$)	3.25000	2.41667	3.66667	3.11111
Modified Exponential ($\delta = 0.0001$)	4.25000	5.00000	4.75000	4.66667
Modified Exponential ($\delta = 0.001$)	4.41667	4.41667	4.00000	4.27778
Modified Exponential ($\delta = 0.01$)	4.25000	2.75000	3.33333	3.44444
Modified Exponential ($\delta = 0.1$)	3.5000	2.75000	3.25000	3.16667

Reinhard et al Global Operator	2.91667	1.66667	2.50000	2.36111
Reinhard et al Local Operator	2.58333	2.00000	3.00000	2.52778
Garrett et al	4.41667	4.50000	4.33333	4.41667
MATLAB “tonemap”	3.41667	2.08333	5.16667	3.55556
Photomatix Details Enhancer	2.33333	2.08333	2.91667	2.44444
Photomatix Tone Compressor	3.50000	1.58333	3.08333	2.72222

Table 9 Ranking of Tone-Mapping Methods for Set 1

Rank	Algorithm	Overall Average Rating
1	Reinhard et al Global Operator	2.36111
2	Photomatix Details Enhancer	2.44444
3	Reinhard et al Local Operator	2.52778
4	Photomatix Tone Compressor	2.72222
5	Exponential ($\delta = 0.01$)	2.80556
6	Exponential ($\delta = 0.001$)	2.91667
7	Exponential ($\delta = 0.1$)	3.11111
8	Exponential ($\delta = 0.0001$)	3.16667
9	Modified Exponential ($\delta = 0.1$)	3.16667
10	Modified Exponential ($\delta = 0.01$)	3.44444
11	MATLAB “tonemap”	3.55556
12	Modified Exponential ($\delta = 0.001$)	4.27778
13	Garrett et al	4.41667
14	Modified Logarithmic	4.52778
15	Modified Exponential ($\delta = 0.0001$)	4.66667
16	Logarithmic	4.88889

4.3.2 Set 2: WDR Image Generation Using MATLAB "makehdr" Funtion Followed by Tone-Mapping Methods

For "book", Reinhard et al local operator shows the best result followed by Reinhard et al global operator, Photomatix tone compressor and exponential (δ=0.0001). They have produced output images which have maximum exposed details and color preservation. (Please refer to **Appendix XX**)

The worst results are shown by modified exponential (δ=0.0001, 0.001, 0.01 and 0.1) and modified logarithmic tone-mapping. Color of images produced by modified exponential (δ=0.0001, 0.001, 0.01 and 0.1) tone-mapping are purplish and unmatched with the original scene. In addition, image details in bright regions are washed-out. Same goes with modified logarithmic tone-mapping but the image is pinkish. Figure 13 shows graphical representation of the computated average rating.

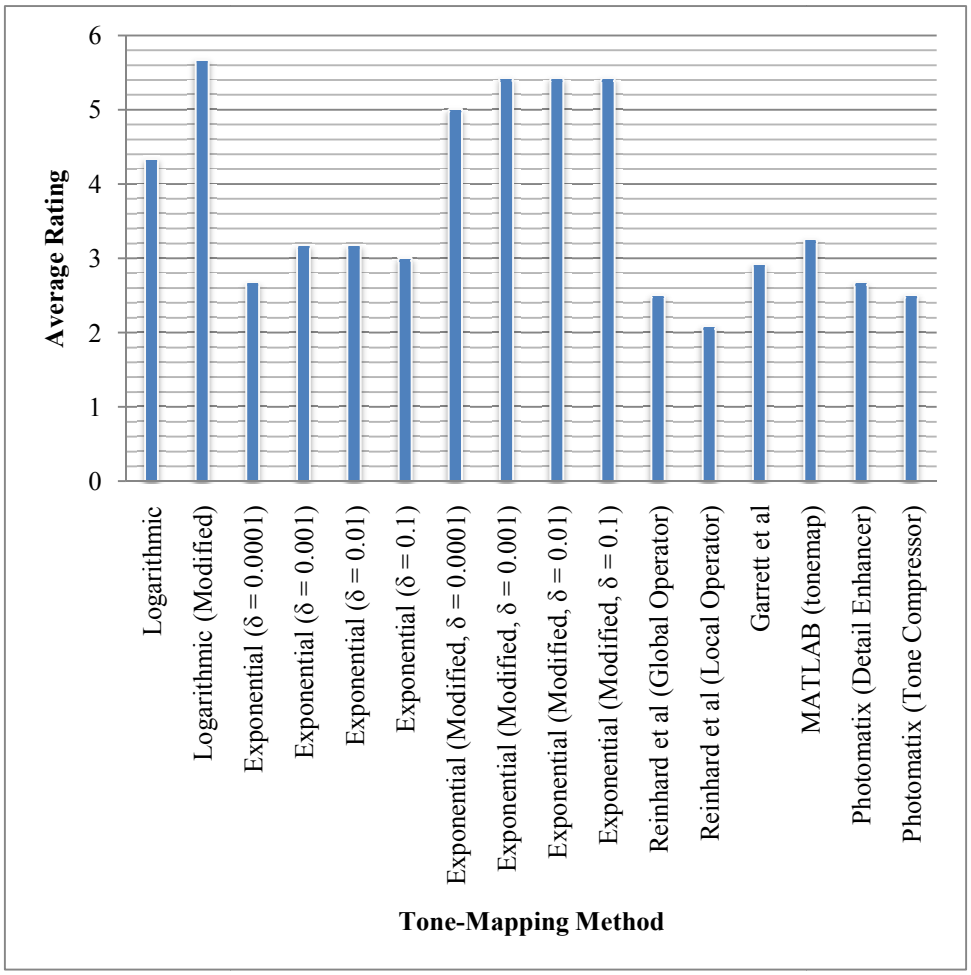


Figure 13 Average Rating of Set 2 (book)

For "lab1", Garrett et al tone-mapping shows the best result followed by Reinhard et al global operator, local operator and MATLAB tone-mapping. Even though Garret et al and MATLAB tone-mapping produced images with slightly less color preservation, but all the best four methods of have resulted images which have well-exposed details. (Please refer to **Appendix XX**)

Like previous evaluation result of "book", the worst results are shown by modified exponential (δ=0.0001, 0.001, 0.01 and 0.1) and modified logarithmic tone-mapping. Color of images produced by modified exponential (δ=0.0001, 0.001, 0.01 and 0.1) tone-mapping are purplish and unmatched with the original scene. In addition, image details in scenes' outdoor are washed-out. Same goes with modified logarithmic tone-mapping but the image is pinkish. Figure 14 shows graphical representation of the computated average rating.

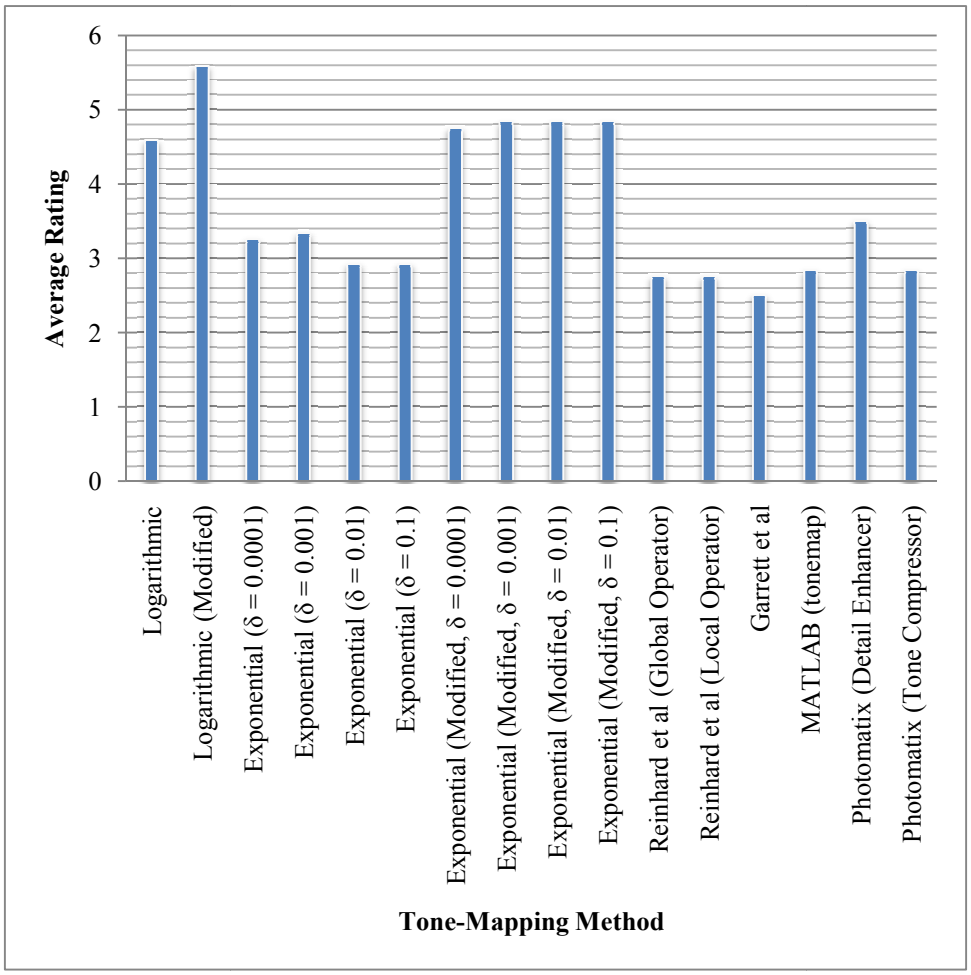


Figure 14 Average Rating of Set 2 (lab1)

For "lab2", Reinhard et al local operator shows the best result followed by Photomatix tone compressor, Reinhard et al global operator and exponential (δ=0.01) tone-mapping. They have produced output images which have moderately-exposed details and color preservation. (Please refer to **Appendix XX**)

Like previous evaluation result of "book", the worst results are shown by modified exponential (δ=0.0001, 0.001, 0.01 and 0.1), logarithmic, and modified logarithmic tone-mapping. Color of images produced by modified exponential (δ=0.0001, 0.001, 0.01 and 0.1) tone-mapping are purplish and unmatched with the original scene. In addition, image details in scene's outdoor are completely washed-out. Same goes with modified logarithmic tone-mapping but the image is

pinkish. For logarithmic tone-mapping, the produced image is dull. Figure 15 shows graphical representation of the computated average rating.

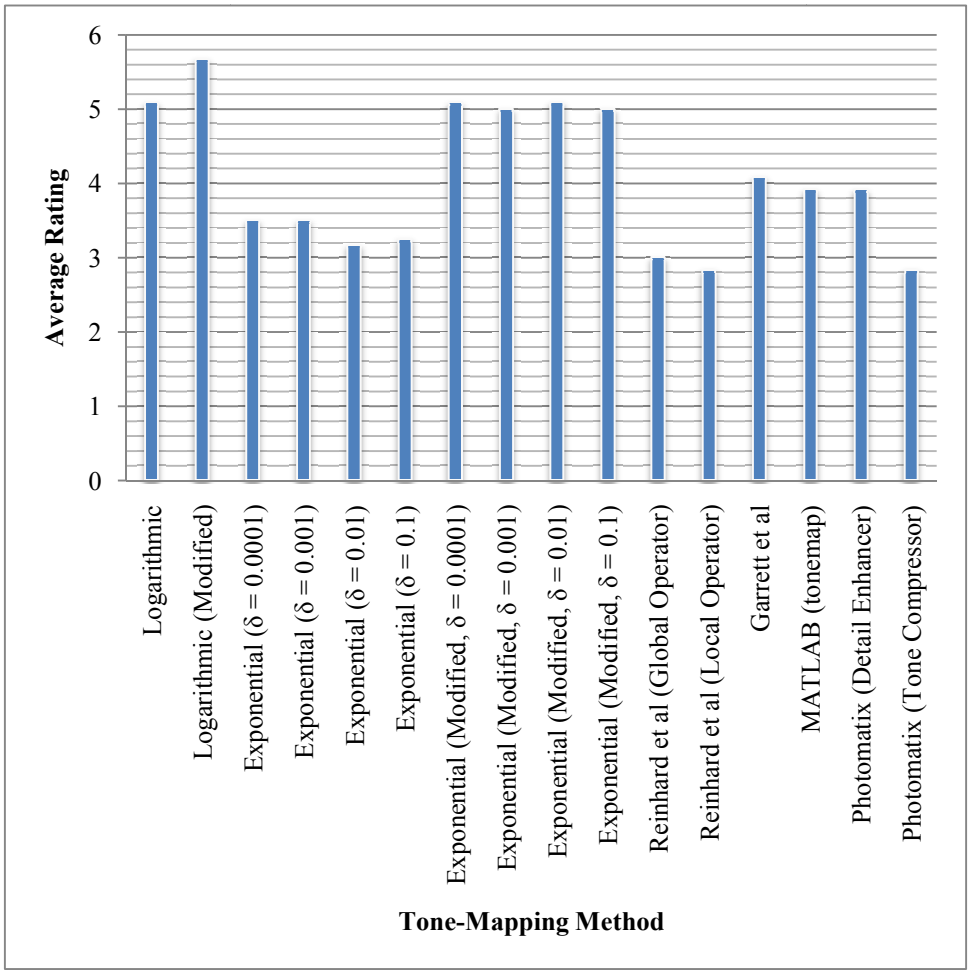


Figure 15 Average Rating of Set 2 (lab2)

For "window", Reinhard et al local operator shows the best result followed by Reinhard et al global operator, exponential (δ=0.01), and Photomatix tone compressor. They have produced output images which have maximum exposed details and color preservation. (Please refer to **Appendix XX**)

The worst results are shown by modified exponential (δ=0.0001 and 0.001) and modified logarithmic tone-mapping. Color of images produced by modified exponential (δ=0.0001 and 0.001) tone-mapping are purplish and not matched with the original scene. In addition, image details in bright regions are washed-out. For modified logarithmic tone-mapping, it produced a dark image. Figure 16 shows graphical representation of the computated average rating.

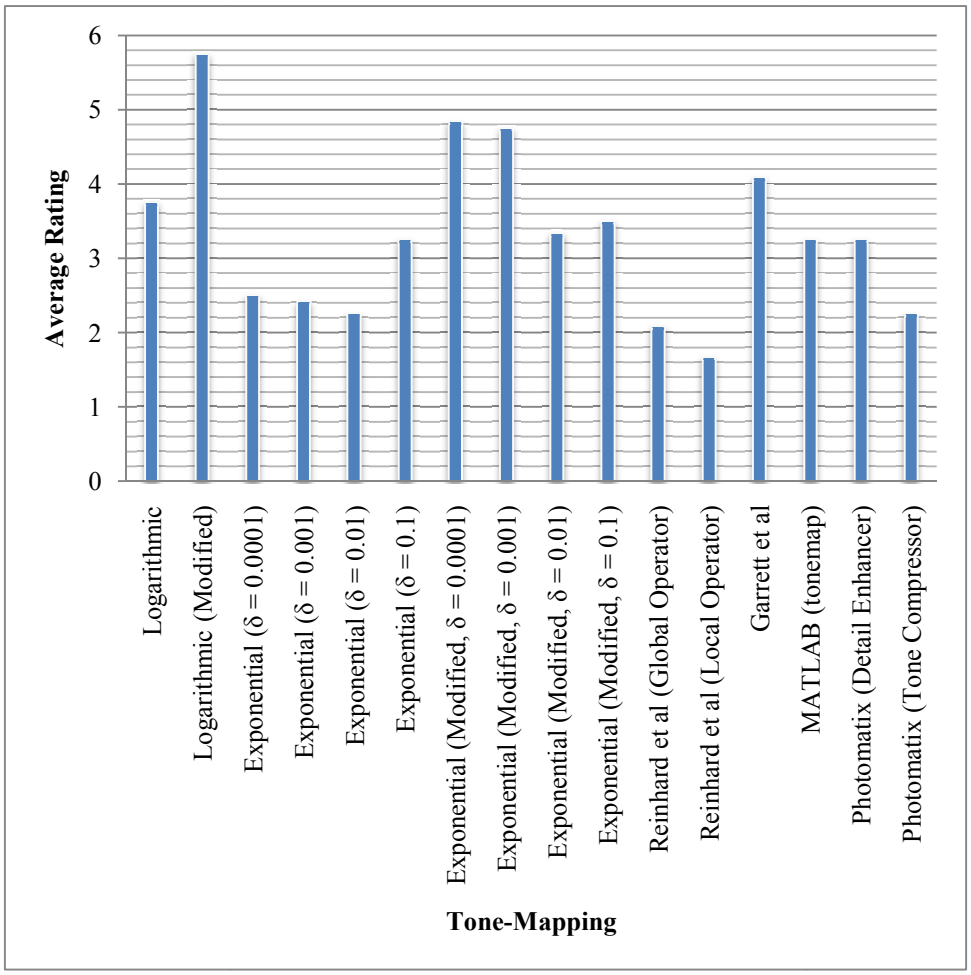


Figure 16 Average Rating of Set 2 (window)

Figure 17 shows graphical representation of the computated average rating for all involved images.

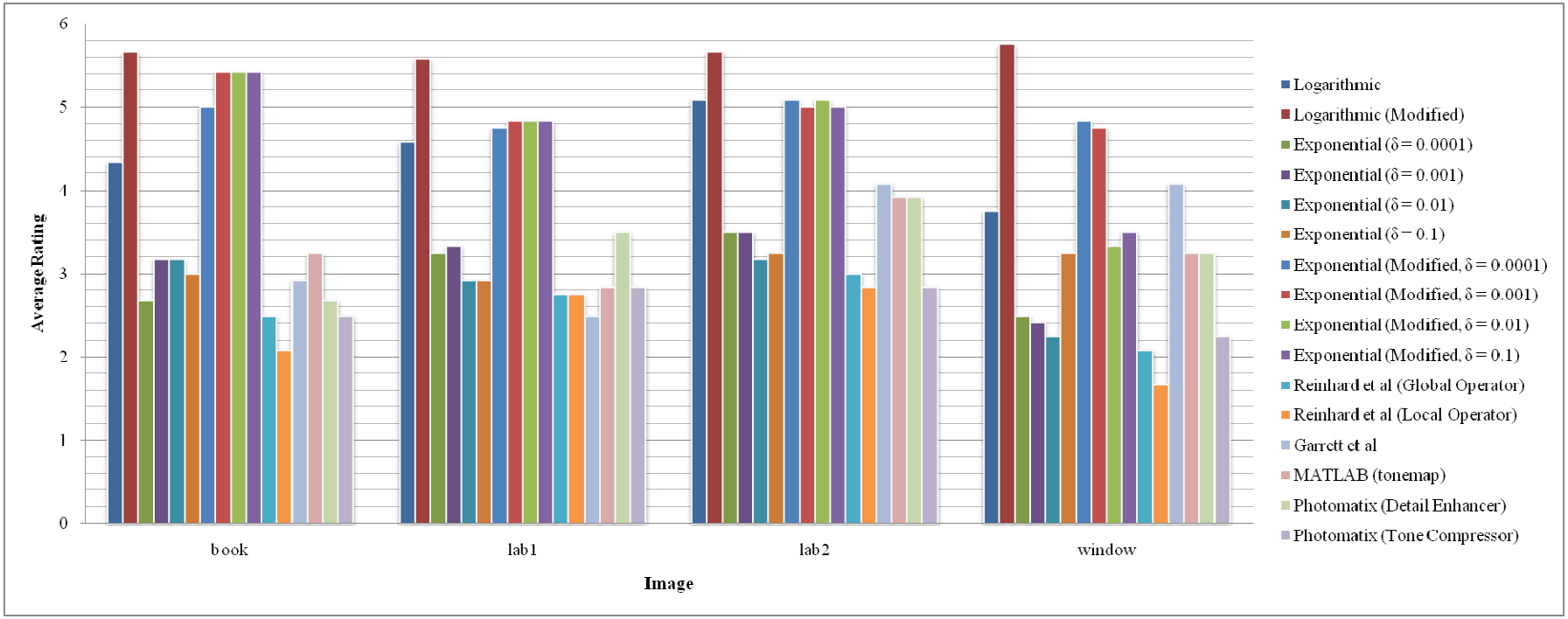


Figure 17 Average Rating of Set 2 (All)

In Table 10, overall average rating are calculated and in Table 11, the algorithms are ranked according their overall average rating. Recall that, smaller rating values means higher quality of images.

Table 10 Overall Average Rating for Set 2

Method	Average Rating				Overall Average Rating
	book	lab1	lab2	window	
Logarithmic	4.33333	4.58333	5.08333	3.75000	4.43750
Modified Logarithmic	5.66667	5.58333	5.66667	5.75000	5.66667
Exponential ($\delta = 0.0001$)	2.66667	3.25000	3.50000	2.50000	2.97917
Exponential ($\delta = 0.001$)	3.16667	3.33333	3.50000	2.41667	3.10417
Exponential ($\delta = 0.01$)	3.16667	2.91667	3.16667	2.25000	2.87500
Exponential ($\delta = 0.1$)	3.00000	2.91667	3.25000	3.25000	3.10417
Modified Exponential ($\delta = 0.0001$)	5.00000	4.75000	5.08333	4.83333	4.91667
Modified Exponential ($\delta = 0.001$)	5.41667	4.83333	5.00000	4.75000	5.00000
Modified Exponential ($\delta = 0.01$)	5.41667	4.83333	5.08333	3.33333	4.66667
Modified Exponential ($\delta = 0.1$)	5.41667	4.83333	5.00000	3.50000	4.68750

Reinhard et al Global Operator	2.50000	2.75000	3.00000	2.08333	2.58333
Reinhard et al Local Operator	2.08333	2.75000	2.83333	1.66667	2.33333
Garrett et al	2.91667	2.50000	4.08333	4.08333	3.39583
MATLAB “tonemap”	3.25000	2.83333	3.91667	3.25000	3.31250
Photomatix Details Enhancer	2.66667	3.50000	3.91667	3.25000	3.33333
Photomatix Tone Compressor	2.50000	2.83333	2.83333	2.25000	2.60417

Table 11 Ranking of Tone-Mapping Methods for Set 2

Rank	Algorithm	Overall Average Rating
1	Reinhard et al Local Operator	2.33333
2	Reinhard et al Global Operator	2.58333
3	Photomatix Tone Compressor	2.60417
4	Exponential ($\delta = 0.01$)	2.87500
5	Exponential ($\delta = 0.0001$)	2.97917
6	Exponential ($\delta = 0.001$)	3.10417
7	Exponential ($\delta = 0.1$)	3.10417
8	MATLAB “tonemap”	3.31250
9	Photomatix Details Enhancer	3.33333
10	Garrett et al	3.39583
11	Logarithmic	4.43750
12	Modified Exponential ($\delta = 0.01$)	4.66667
13	Modified Exponential ($\delta = 0.1$)	4.68750
14	Modified Exponential ($\delta = 0.0001$)	4.91667
15	Modified Exponential ($\delta = 0.001$)	5.00000
16	Modified Logarithmic	5.66667

4.3.3 Set 3: WDR Image Generation Using P. Debevec and J. Malik Method Followed by Tone-Mapping Methods

For "book", exponential (δ=0.1) shows the best result followed by exponential (δ=0.0001), exponential (δ=0.001), exponential (δ=0.01), Photomatix tone compressor and Reinhard et al global operator. They have produced output images which have maximum exposed details and color preservation. (Please refer to **Appendix YY**)

The worst results are shown by modified exponential (δ=0.0001, 0.001, 0.01 and 0.1). Color of images produced by modified exponential (δ=0.0001, 0.001, 0.01 and 0.1) tone-mapping are purplish and not matched with the original scene. In addition, image details in bright regions are washed-out. Figure 18 shows graphical representation of the computated average rating.

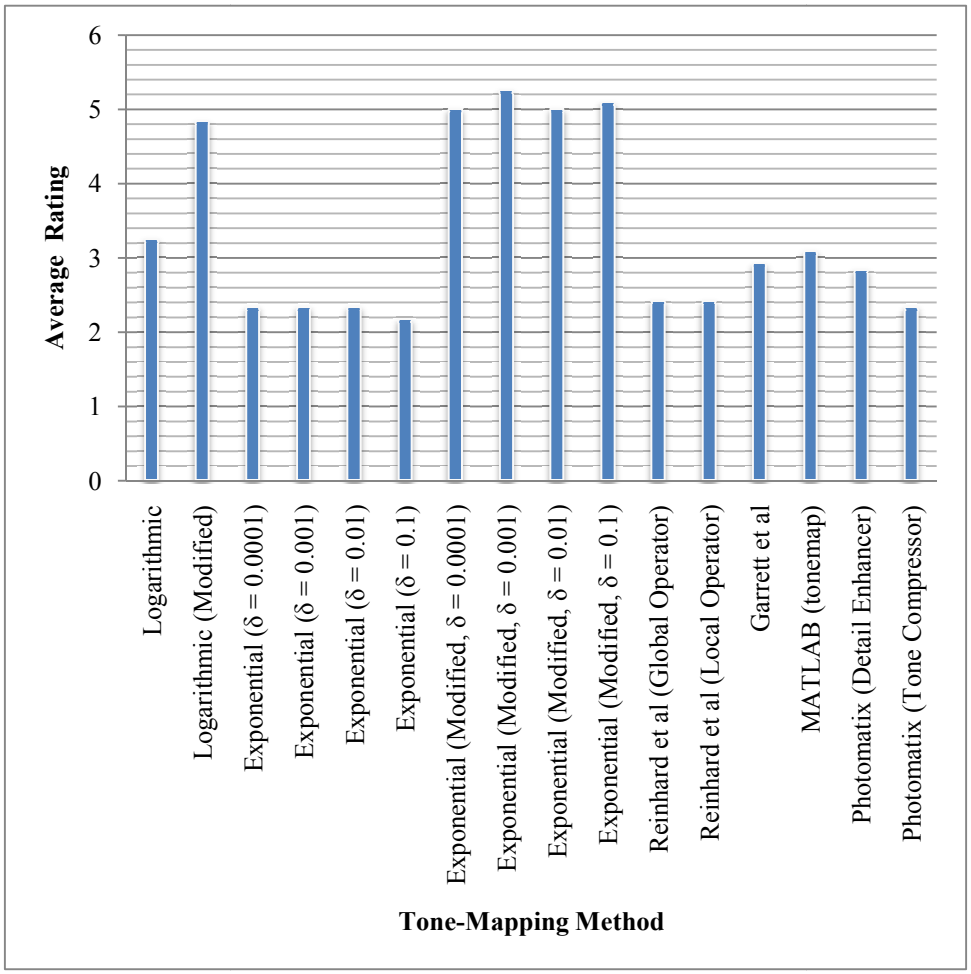


Figure 18 Average Rating of Set 3 (book)

For “lab1”, Photomatix detail enhancer shows the best result followed by MATLAB “tonemap” and Photomatix tone compressor. They have produced output images which have well-exposed details and color preservation. (Please refer to **Appendix YY**)

The worst results are shown by modified exponential (δ=0.0001 and 0.001) and Garrett et al tone-mapping. Color of images produced by modified exponential (δ=0.0001 and 0.001) tone-mapping are purplish and not matched with the original scene. In addition, image is somewhat posterized where there are abrupt changes from one tone to another and details in bright regions are washed-out. For Garrett et al tone-mapping, the output images is dark and details in dark regions are under-exposed. Figure 19 shows graphical representation of the computated average rating.

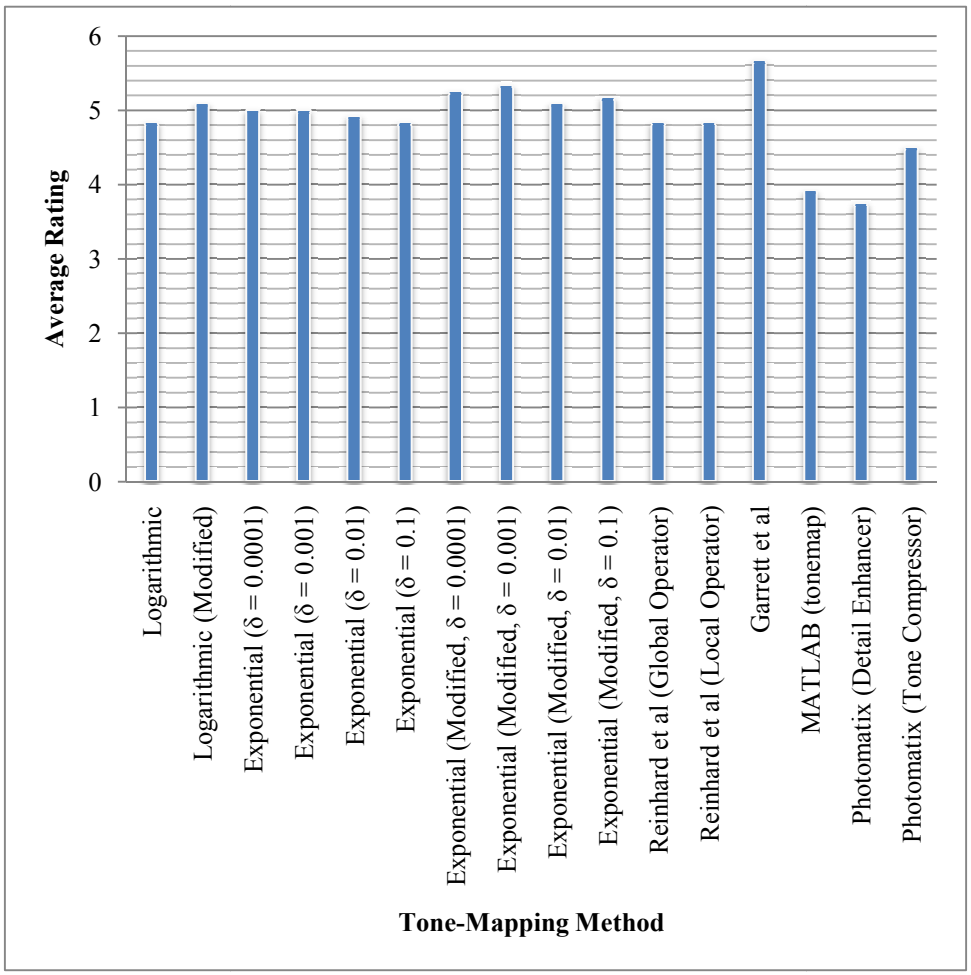


Figure 19 Average Rating of Set 3 (lab1)

For “lab2”, Photomatix detail enhancer shows the best result followed by logarithmic and Photomatix tone compressor. They have produced output images which have well-exposed details and color preservation. (Please refer to **Appendix YY**)

Like previous evaluation result of “lab1”, the worst results are shown by modified exponential (δ=0.0001 and 0.001) and Garrett et al tone-mapping. Color of images produced by modified exponential (δ=0.0001 and 0.001) tone-mapping are purplish and not matched with the original scene. In addition, image is somewhat posterized where there are slightly abrupt changes from one tone to another and details of scene’s outdoor are completely washed-out. For Garrett et al tone-mapping, the output images is dark and details in dark regions are under-exposed. Figure 20 shows graphical representation of the computated average rating.

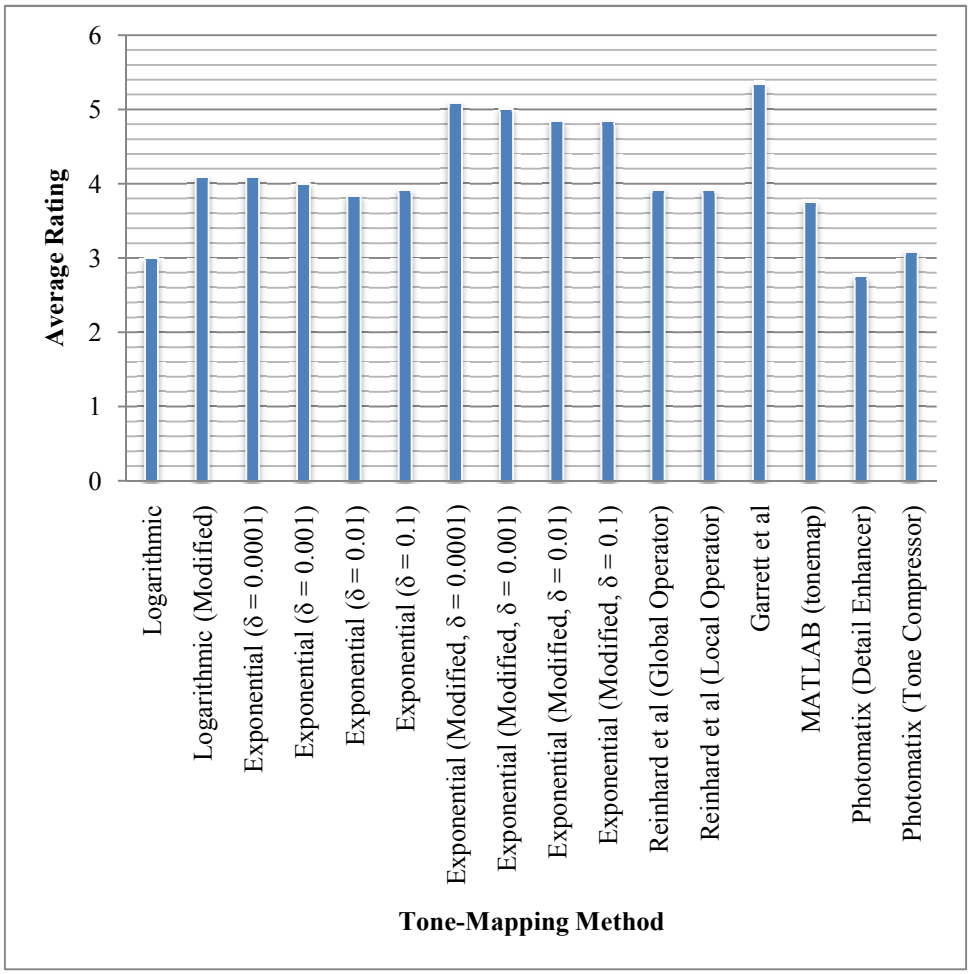


Figure 20 Average Rating of Set 3 (lab2)

For “window”, Reinhard et al global operator shows the best result followed by exponential (δ=0.0001, 0.001, 0.01 and 0.1) and Photomatix tone compressor. They have produced output images which have well-exposed details and color preservation. (Please refer to **Appendix YY**)

The worst results are shown by modified exponential (δ=0.0001, 0.001 and 0.01). Color of images produced by modified exponential (δ=0.0001, 0.001 and 0.01) tone-mapping are not matched with the original scene. In addition, details in bright regions are washed-out. Figure 21 shows graphical representation of the computated average rating.

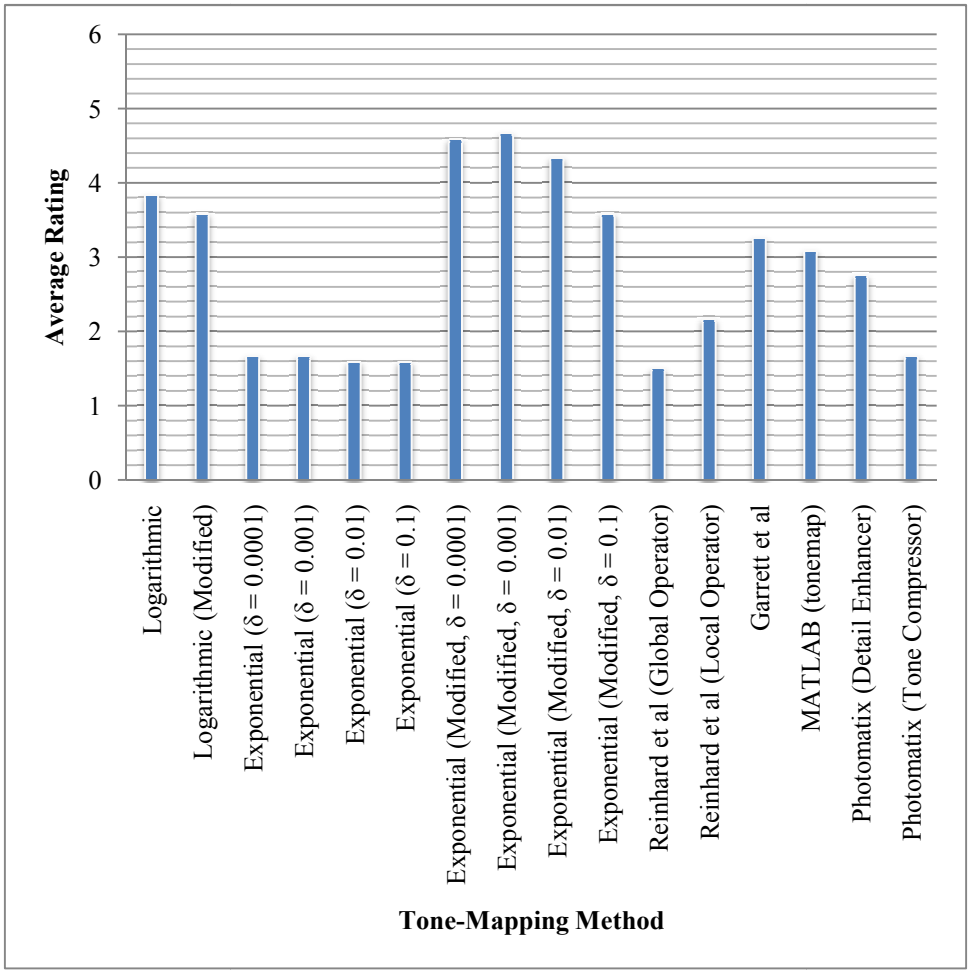


Figure 21 Average Rating of Set 3 (window)

Figure 22 shows graphical representation of the computated average rating for all involved images.

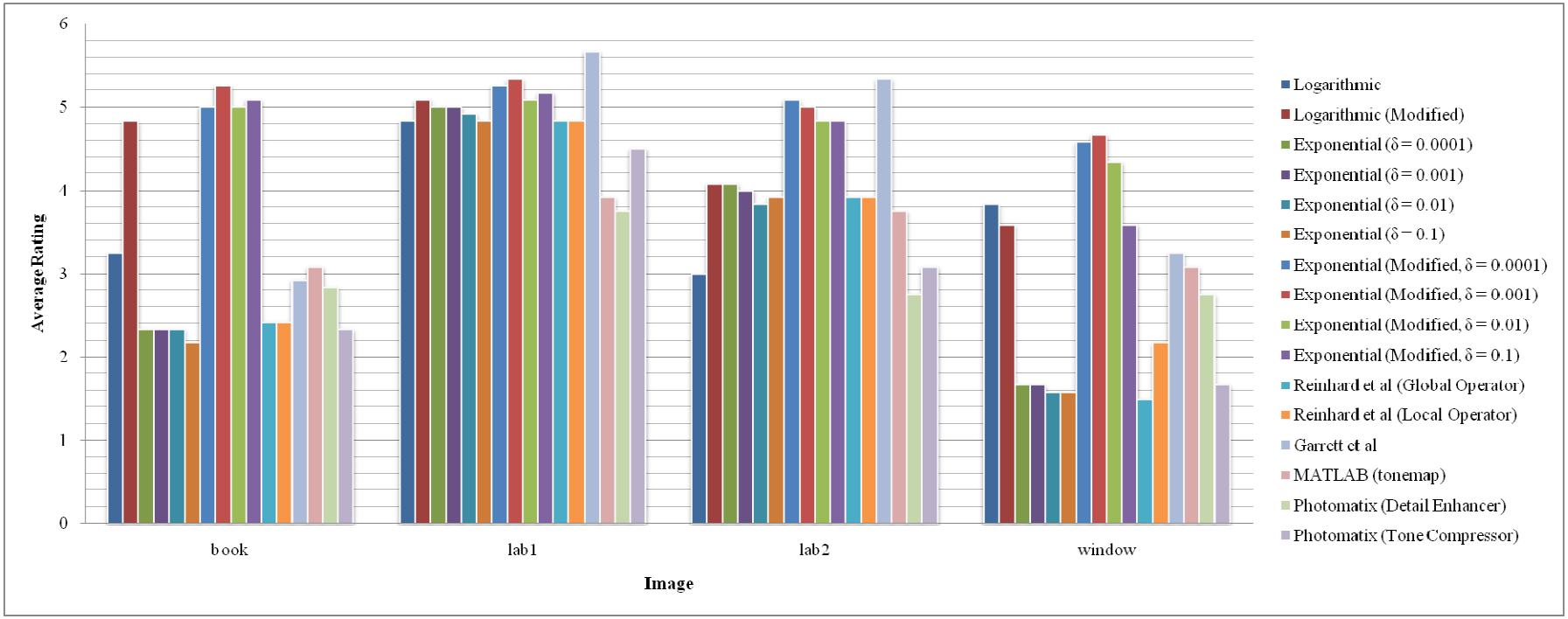


Figure 22 Average Rating of Set 3 (All)

In Table 12, overall average rating are calculated and in Table 13, the algorithms are ranked according their overall average rating. Recall that, smaller rating values means higher quality of images.

Table 12 Overall Average Rating for Set 3

Method	Average Rating				Overall Average Rating
	book	lab1	lab2	window	
Logarithmic	3.25000	4.83333	3.00000	3.83333	3.72917
Modified Logarithmic	4.83333	5.08333	4.08333	3.58333	4.39583
Exponential ($\delta = 0.0001$)	2.33333	5.00000	4.08333	1.66667	3.27083
Exponential ($\delta = 0.001$)	2.33333	5.00000	4.00000	1.66667	3.25000
Exponential ($\delta = 0.01$)	2.33333	4.91667	3.83333	1.58333	3.16667
Exponential ($\delta = 0.1$)	2.16667	4.83333	3.91667	1.58333	3.12500
Modified Exponential ($\delta = 0.0001$)	5.00000	5.25000	5.08333	4.58333	4.97917
Modified Exponential ($\delta = 0.001$)	5.25000	5.33333	5.00000	4.66667	5.06250
Modified Exponential ($\delta = 0.01$)	5.00000	5.08333	4.83333	4.33333	4.81250
Modified Exponential ($\delta = 0.1$)	5.08333	5.16667	4.83333	3.58333	4.66667

Reinhard et al Global Operator	2.41667	4.83333	3.91667	1.50000	3.16667
Reinhard et al Local Operator	2.41667	4.83333	3.91667	2.16667	3.33333
Garrett et al	2.91667	5.66667	5.33333	3.25000	4.29167
MATLAB “tonemap”	3.08333	3.91667	3.75000	3.08333	3.45833
Photomatix Details Enhancer	2.83333	3.75000	2.75000	2.75000	3.02083
Photomatix Tone Compressor	2.33333	4.50000	3.08333	1.66667	2.89583

Table 13 Ranking of Tone-Mapping Methods for Set 3

Rank	Algorithm	Overall Average Rating
1	Photomatix Tone Compressor	2.89583
2	Photomatix Details Enhancer	3.02083
3	Exponential ($\delta = 0.1$)	3.12500
4	Exponential ($\delta = 0.01$)	3.16667
5	Reinhard et al Global Operator	3.16667
6	Exponential ($\delta = 0.001$)	3.25000
7	Exponential ($\delta = 0.0001$)	3.27083
8	Reinhard et al Local Operator	3.33333
9	MATLAB “tonemap”	3.45833
10	Logarithmic	3.72917
11	Garrett et al	4.29167
12	Modified Logarithmic	4.39583
13	Modified Exponential ($\delta = 0.1$)	4.66667
14	Modified Exponential ($\delta = 0.01$)	4.81250
15	Modified Exponential ($\delta = 0.0001$)	4.97917
16	Modified Exponential ($\delta = 0.001$)	5.06250

4.3.4 Set 4: WDR Image Generation Using Photomatix WDR Generator Followed by Tone-Mapping Methods

For "book", Reinhard et al local operator shows the best result followed by exponential (δ=0.01), Reinhard et al global operator and exponential (δ=0.001). They have produced output images which have maximum exposed details and color preservation. (Please refer to **Appendix ZZ**)

The worst results are shown by modified exponential (δ=0.0001 and 0.001) and logarithmic tone-mapping. Color of images produced by modified exponential (δ=0.0001 and 0.001) tone-mapping are purplish and not matched with the original scene. In addition, details in bright regions are washed-out. For logarithmic tone-mapping, the output image is quite dark. Figure 23 shows graphical representation of the computated average rating.

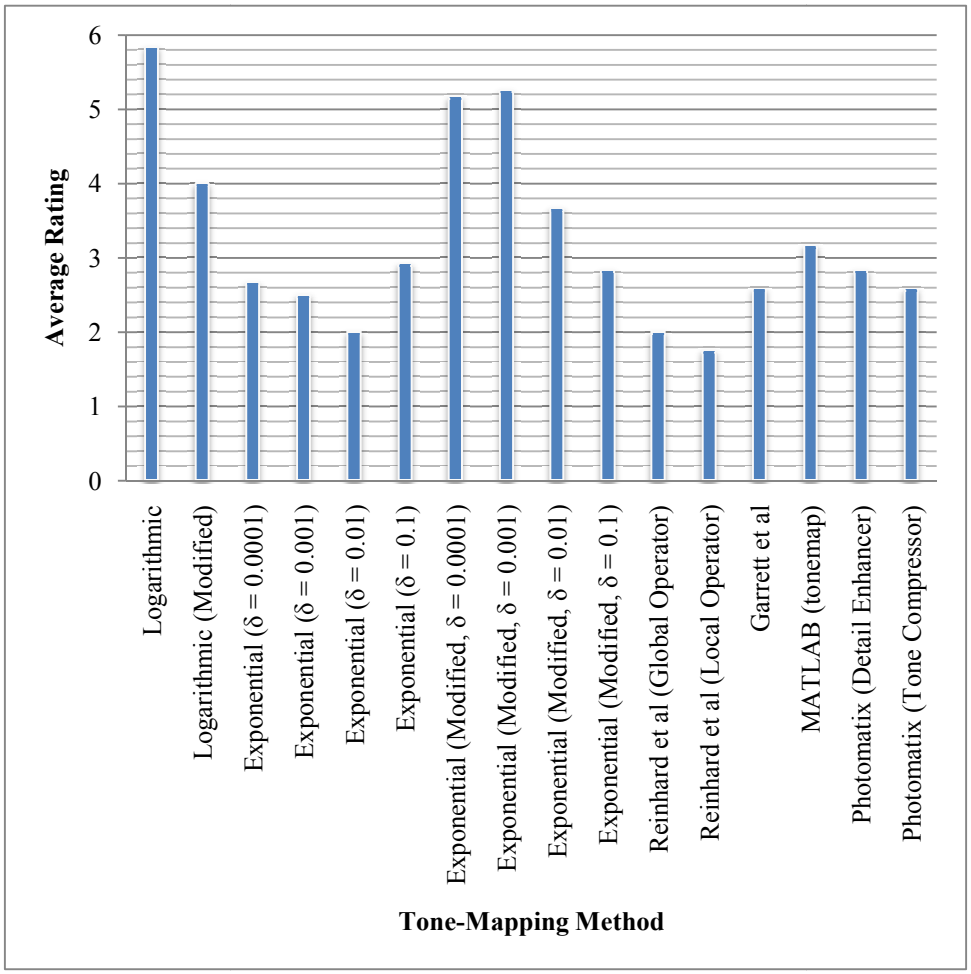


Figure 23 Average Rating of Set 4 (book)

For “lab1”, Reinhard et al global operator shows the best result followed by Reinhard et al local operator, exponential (δ=0.01), and exponential (δ=0.001). They have produced output images which have maximum exposed details and color preservation. (Please refer to **Appendix ZZ**)

The worst results are shown by Garret et al, logarithmic, modified logarithmic, modified exponential (δ=0.0001 and 0.001) tone-mapping. Color of images produced by modified exponential (δ=0.0001 and 0.001) tone-mapping are purplish and not matched with the original scene. In addition, details of scene’s outdoor are washed-out. For Garrett et al, logarithmic and modified logarithmic tone-mapping, their output images are quite dark there are losses of details in dark regions. Figure 24 shows graphical representation of the computated average rating.

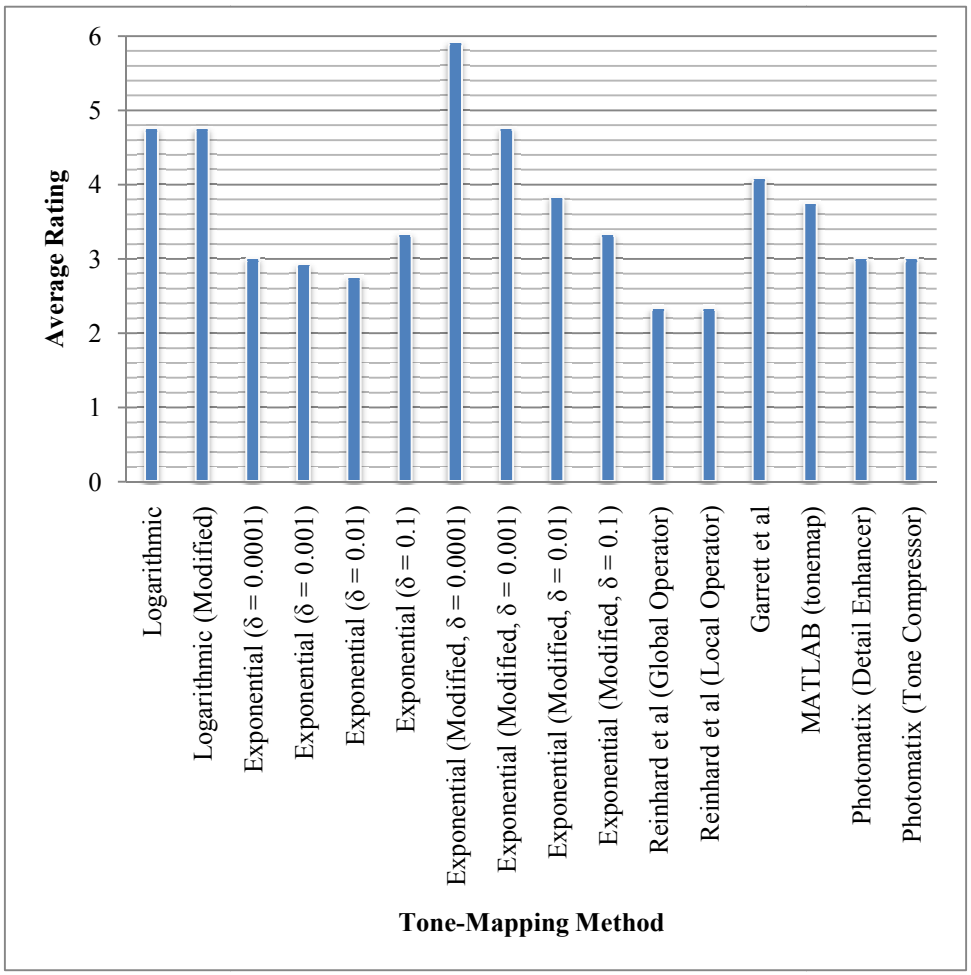


Figure 24 Average Rating of Set 4 (lab1)

For “lab2”, Reinhard et al local operator shows the best result followed by Reinhard et al global operator, and exponential (δ=0.01). Both Reinhard et al local and global operator have produced images which have well-exposed details and color preservation. Even though exponential (δ=0.01) produce image with some losses of details of scene’s outdoor, the output image is quite good from the observers’ point of view. (Please refer to **Appendix ZZ**)

The worst results are shown by modified logarithmic, logarithmic, and modified exponential (δ=0.0001) tone-mapping. Color of images produced by modified exponential (δ=0.0001) tone-mapping are purplish and not matched with the original scene. In addition, details of scene’s outdoor are completely washed-out. For modified logarithmic and logarithmic tone-mapping, their output images are quite dark there are losses of details in dark regions. Figure 25 shows graphical representation of the computated average rating.

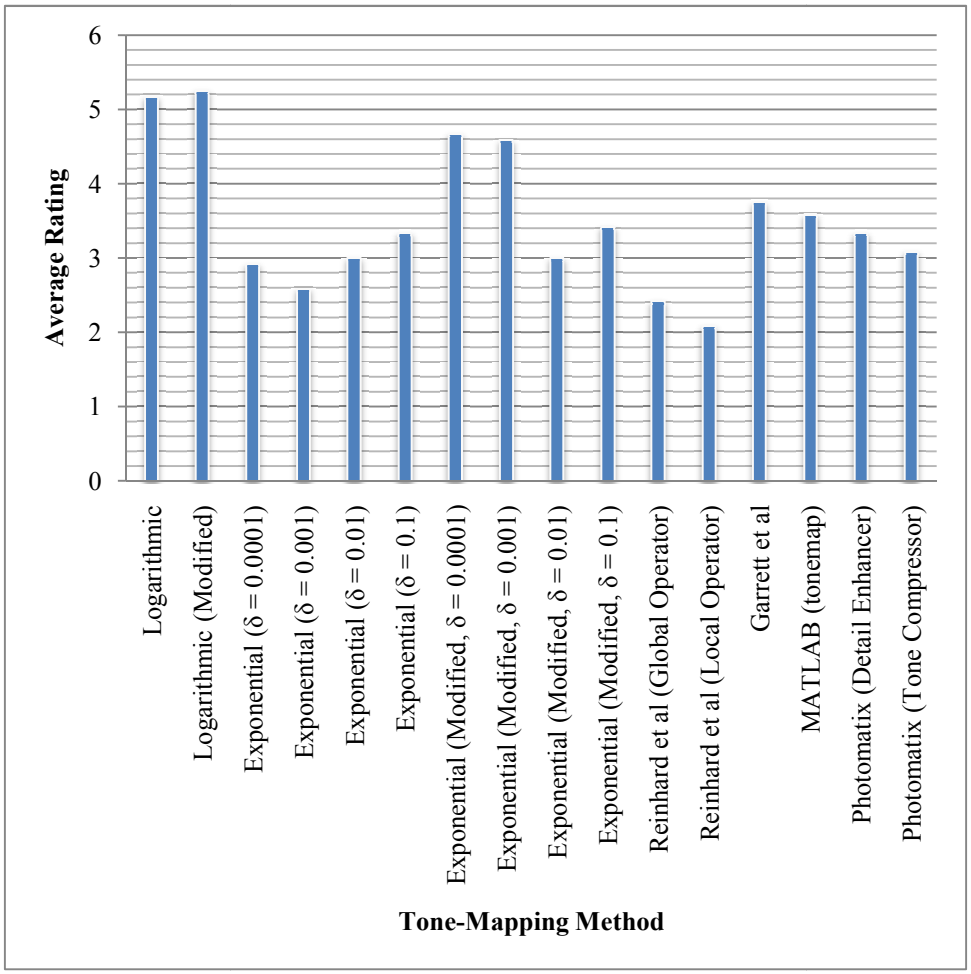


Figure 25 Average Rating of Set 4 (lab2)

For “window”, exponential (δ=0.001) shows the best result followed by, exponential (δ=0.0001) and Reinhard et al global operator. They have produced output images which have maximum exposed details and color preservation. (Please refer to **Appendix ZZ**)

The worst results are shown by modified exponential (δ=0.0001 and 0.1), logarithmic and modified logarithmic tone-mapping. Color of image produced by modified exponential (δ=0.0001) tone-mapping is not matched with the original scene. In addition, details in bright regions are washed-out. For exponential (δ=0.1), logarithmic and modified logarithmic tone-mapping, the output image is quite dark. Figure 26 shows graphical representation of the computated average rating.

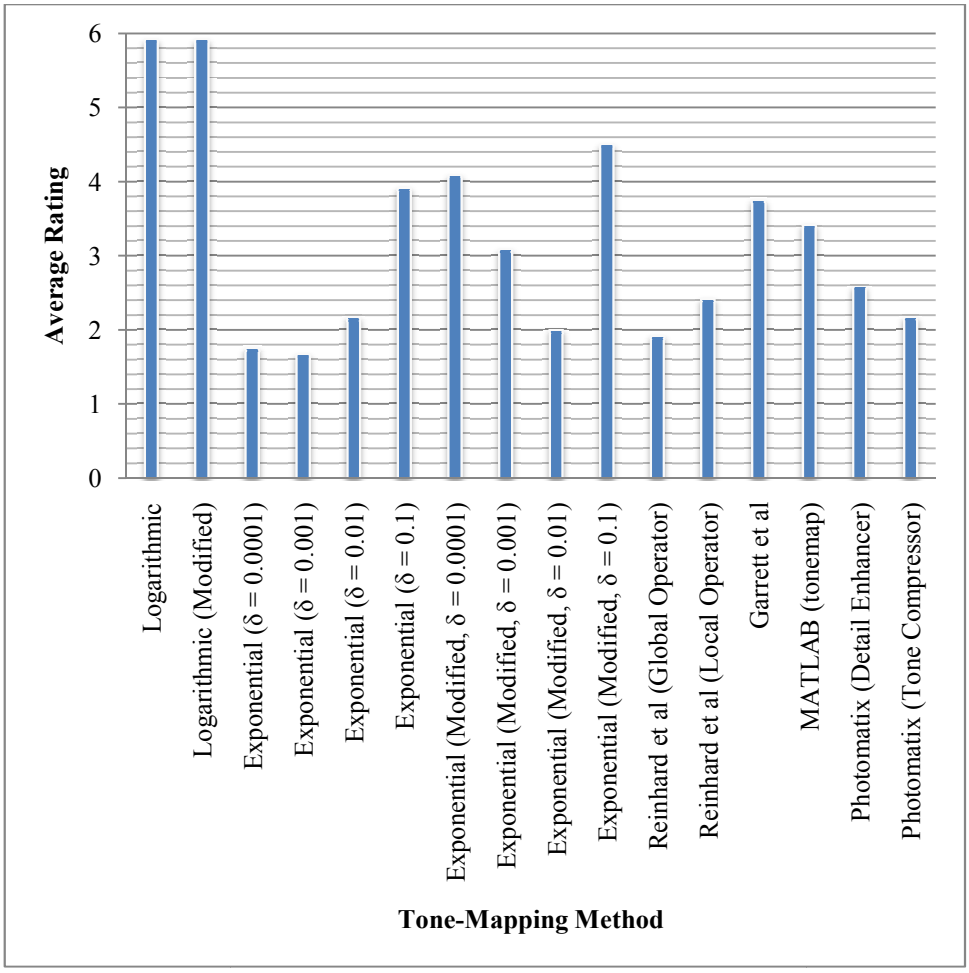


Figure 26 Average Rating of Set 4 (window)

Figure 27 shows graphical representation of the computated average rating for all involved images.

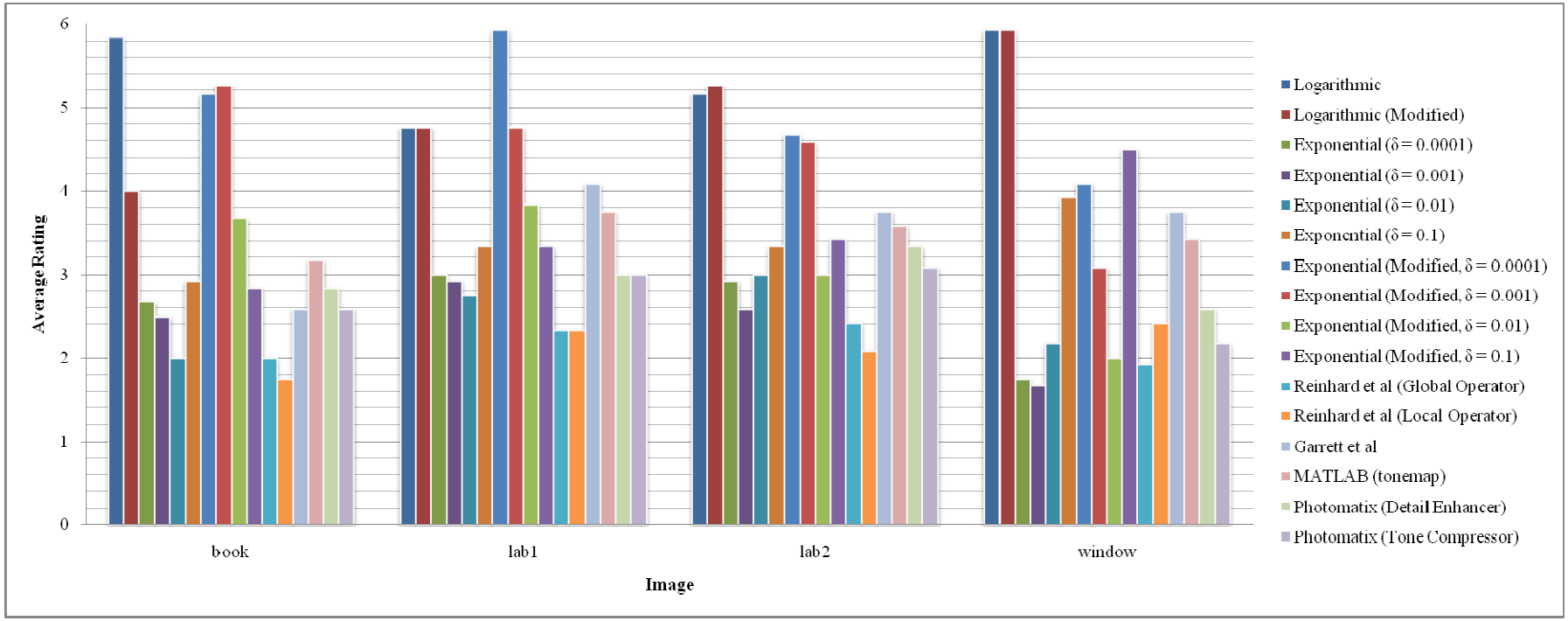


Figure 27 Average Rating of Set 4 (All)

In Table 14, overall average rating are calculated and in Table 15, the algorithms are ranked according their overall average rating. Recall that, smaller rating values means higher quality of images.

Table 14 Overall Average Rating for Set 4

Method	Average Rating				Overall Average Rating
	book	lab1	lab2	window	
Logarithmic	5.83333	4.75000	5.16667	5.91667	5.41667
Modified Logarithmic	4.00000	4.75000	5.25000	5.91667	4.97917
Exponential ($\delta = 0.0001$)	2.66667	3.00000	2.91667	1.75000	2.58334
Exponential ($\delta = 0.001$)	2.50000	2.91667	2.58333	1.66667	2.41667
Exponential ($\delta = 0.01$)	2.00000	2.75000	3.00000	2.16667	2.47917
Exponential ($\delta = 0.1$)	2.91667	3.33333	3.33333	3.91667	3.37500
Modified Exponential ($\delta = 0.0001$)	5.16667	5.91667	4.66667	4.08333	4.95834
Modified Exponential ($\delta = 0.001$)	5.25000	4.75000	4.58333	3.08333	4.41667
Modified Exponential ($\delta = 0.01$)	3.66667	3.83333	3.00000	2.00000	3.12500
Modified Exponential ($\delta = 0.1$)	2.83333	3.33333	3.41667	4.50000	3.52083

Reinhard et al Global Operator	2.00000	2.33333	2.41667	1.91667	2.16667
Reinhard et al Local Operator	1.75000	2.33333	2.08333	2.41667	2.14583
Garrett et al	2.58333	4.08333	3.75000	3.75000	3.54167
MATLAB “tonemap”	3.16667	3.75000	3.58333	3.41667	3.47917
Photomatix Details Enhancer	2.83333	3.00000	3.33333	2.58333	2.93750
Photomatix Tone Compressor	2.58333	3.00000	3.08333	2.16667	2.70833

Table 15 Ranking of Tone-Mapping Methods for Set 4

Rank	Algorithm	Overall Average Rating
1	Reinhard et al Local Operator	2.14583
2	Reinhard et al Global Operator	2.16667
3	Exponential ($\delta = 0.001$)	2.41667
4	Exponential ($\delta = 0.01$)	2.47917
5	Exponential ($\delta = 0.0001$)	2.58334
6	Photomatix Tone Compressor	2.70833
7	Photomatix Details Enhancer	2.93750
8	Modified Exponential ($\delta = 0.01$)	3.12500
9	Exponential ($\delta = 0.1$)	3.37500
10	MATLAB “tonemap”	3.47917
11	Modified Exponential ($\delta = 0.1$)	3.52083
12	Garrett et al	3.54167
13	Modified Exponential ($\delta = 0.001$)	4.41667
14	Modified Exponential ($\delta = 0.0001$)	4.95834
15	Modified Logarithmic	4.97917
16	Logarithmic	5.41667

4.3.5 Performance of All Methods

Now, the performance of tone-mapping methods across the WDR image generation methods is evaluated as shown in Table 56. For each set, the best- and the worst-two tone-mapping methods are indicated with **green** and **orange** color.

For WDR image generation, Photoatix WDR generator gives the best results followed by MATLAB "makehdr" function and P. Debevec and J. Malik method, where their best rating is 2.14583, 2.33333 and 2.89583, respectively.

For tone-mapping, the best method is Reinhard et al local operator, followed by Reinhard et al global operator where their best rating is 2.14583 and 2.16667, respectively. On the other hand, the method which gives worst result is modified logarithmic tone-mapping followed by logarithmic tone-mapping where their worst rating is 5.66667 and 5.41667, respectively.

Table 16 Overall Average Rating for All Sets

Tone-Mapping Method	Overall Average Rating			
	Set 1	Set 2	Set 3	Set 4
Logarithmic	**4.88889**	4.43750	3.72917	**5.41667**
Modified Logarithmic	4.52778	**5.66667**	4.39583	**4.97917**
Exponential ($\delta = 0.0001$)	3.16667	2.97917	3.27083	2.58334
Exponential ($\delta = 0.001$)	2.91667	3.10417	3.25000	2.41667
Exponential ($\delta = 0.01$)	2.80556	2.87500	3.16667	2.47917
Exponential ($\delta = 0.1$)	3.11111	3.10417	3.12500	3.37500
Modified Exponential ($\delta = 0.0001$)	**4.66667**	4.91667	4.97917	4.95834

Modified Exponential ($\delta = 0.001$)	4.27778	**5.00000**	5.06250	4.41667
Modified Exponential ($\delta = 0.01$)	3.44444	4.66667	4.81250	3.12500
Modified Exponential ($\delta = 0.1$)	3.16667	4.68750	4.66667	3.52083
Reinhard et al Global Operator	**2.36111**	**2.58333**	3.16667	**2.16667**
Reinhard et al Local Operator	2.52778	**2.33333**	3.33333	**2.14583**
Garrett et al	4.41667	3.39583	4.29167	3.54167
MATLAB "tonemap"	3.55556	3.31250	3.45833	3.47917
Photomatix Details Enhancer	**2.44444**	3.33333	**3.02083**	2.93750
Photomatix Tone Compressor	2.72222	2.60417	**2.89583**	2.70833

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

For WDR image generation, Photomatix WDR generator gives the best results followed by MATLAB "makehdr" function and P. Debevec and J. Malik method. For tone-mapping, the best method is Reinhard et al local operator, followed by Reinhard et al global operator. On the other hand, the method which gives worst result is modified logarithmic tone-mapping followed by logarithmic tone-mapping. When compromising between quality and speed, Reinhard et al global operator performs the best.

Conclusively, each tone-mapping method has different ability to deal with different WDR images. Their performances are different in preserving details, brightness, contrast, and color appearance. In addition, by using different methods of WDR image generation, it affects the output images resulted from same tone-mapping method. Of course, we desire to produce good quality of images and try to manipulate the parameters in the algorithms to achieve it. Besides, the complexity of method and image size affect the computation time. Also, from the results of observers' evaluation, the performances of all combination of chosen WDR image generation and tone-mapping methods had been successfully identified.

5.2 RECOMMENDATION

It is suggested that latest WDR imaging methods should be explored and implemented. It is beneficial to conduct projects which require implementations and analysis since we are continuously want to test the reliability of the developed methods or algorithms. The works done and data collected will be useful to provide a base for any future image processing project.

REFERENCES

[1] Linan Jiao, Zhaoyun Sun, Aimin Sha, & Juan Zhang, 2009, "Adaptive Local Contrast Enhancement of High Dynamic Range Image"

[2] Jian Zhang & Sei-ichiro Kamata, 2008, "Adaptive Local Contrast Enhancement for the Visualization of High Dynamic Range Images"

[3] Chun Hung Liu, Oscar C. Au, P. H. W. Wong, & M. C. Kung, 2008, "Image Characteristic Oriented Tone Mapping for High Dynamic Range Images"

[4] Erik Reinhard, Michael Stark, Peter Shirley, & James Ferweda, 2002, "Photographic Tone Reproduction for Digital Images"

[5] *High Dynamic Range Imaging*, Wikipedia.com. Retrieved February 4, 2010, from http://en.wikipedia.org/wiki/High_dynamic_range_imaging

[6] Reinhard, E., Ward, G., Pattanaik, S., & Debevec, P., 2006, *High Dynamic Range Imaging: Acquisition, Display and Image-based Lighting*. United Kingdom, UK: Morgan Kaufmann Publishers

[7] *Understanding Dynamic Range in Digital Photography*, CambridgeInColour.com. Retrieved March 31, 2010, from http://www.cambridgeincolour.com/tutorials/dynamic-range.htm

[8] *Wide Dynamic Range*, Wikipedia.com. Retrieved January 26, 2010, from http://en.wikipedia.org/wiki/Wide_dynamic_range

[9] Paul E. Debevec & Jitendra Malik, 1997, "Recovering High Dynamic Range Radiance Maps from Photographs"

[10] Wen-Chung Kao, Wei-Chi Huang, Lien-Yang Chen, Hong-Shuo Tai & Hung-Hsin Wu, 2008, "Taking Images in Extremely High Dynamic Range Scenes by Fusing Multiple Exposed Images and Tone Reproduction"

[11] Wen-Chung Kao, 2008, "High Dynamic Range Imaging by Fusing Multiple Raw Images and Tone Reproduction"

[12] I. R. Khan, Z. Huang, F. Farbiz, & C. M. Manders, 2009, "HDR Image Tone Mapping Using Histogram Adjustment Adapted to Human Visual System"

[13] Jianbeng Shen, Hanqiu Sun, Hanli Zhao, & Xiaogang Jin, 2009, "RAM-Based Tone Mapping for High Dynamic Range Images"

[14] *LCD Color: 8-bit vs. 6-bit*, CompReviews.com. Retrieved February 4, 2010, from http://compreviews.about.com/od/multimedia/a/LCDColor.htm

[15] Gonzales, R. C., & Woods R. E., 2010, *Digital Image Processing, Third* Edition. United State of America, USA: Pearson Prentice Hall.

[16] *HDR Images in Photography – About Dynamic Range, Tone Mapping and HDR Imaging for Photography*, HDRsoft.com. Retrieved March 27, 2010, from http://www.hdrsoft.com/resources/dri.html

[17] Won-Ho Cho & Ki-Sang Hong, 2008, "Extending Dynamic Range of Two Color Images under Different Exposures"

[18] Tzern-Ru Chou, 2008, "Color Calibration of Recovering High Dynamic Range Images"

[19] *Create High Dynamic Range Image – MATLAB*, MathWorks.com. Retrieved March 31, 2010, from http://www.mathworks.com/access/helpdesk/help/toolbox/images/makehdr.html

[20] André A. Bell, Claude Seiler, Jens N. Kaftan, & Til Aach, 2008, "Noise in High Dynamic Range Imaging"

[21] Neil Barakat, Thomas E. Darcie, & A. Nicholas Hone, 2008, "The Tradeoff between SNR and Exposure-Set Size in HDR Imaging"

[22] Takao Jinno & Masahiro Okuda, 2008, "Motion Blur Free HDR Image Acquisition Using Multiple Exposures"

[23] Tae-Hong Min, Rae-Hong Park, & Soon-Koon Chang, 2009, "Histogram Based Ghost Removal in High Dynamic Range Images"

[24] Ansel Adams, 1980, *The Camera: The Ansel Adams Photography Series.* United State of America, USA: Little Brown and Company

[25] Ansel Adams, 1980, *The Negative: The Ansel Adams Photography Series.* United State of America, USA: Little Brown and Company

[26] Ansel Adams, 1980, *The Print: The Ansel Adams Photography Series*. United State of America, USA: Little Brown and Company

[27] Zhongkang Lu & Susanto Rahardja, 2009, "Realistic HDR Tone-Mapping Based on Contrast Perception Matching"

[28] Garrett M. Johnson & Mark D. Fairchild, 2005, "Rendering HDR Images"

[29] Nathan Moroney, Mark D. Fairchild, Robert W. G. Hunt, M. Ronnier Luo, & Todd Newman, 2002, "The CIECAM02 Color Appearance Model"

[30] *Render High Dynamic Range Image for Viewing – MATLAB*, MathWorks.com. Retrieved March 31, 2010, from http://www.mathworks.com/access/helpdesk/help/toolbox/images/tonemap.html

[31] *HDR Photo Software & Plugin for Lightroom, Aperture, & Photoshop – Tone Mapping, Exposure Fusion & HDR Imaging for Photography*, HDRsoft.com. Retrieved January 31, 2010, from http://www.hdrsoft.com/

[32] *Photomatix User Gallery*, HDRsoft.com. Retrieved January 31, 2010, from http://www.hdrsoft.com/gallery/index.php

[33] *Photographic Tone Reproduction for Digital Images*, CS.Utah.edu. Retrieved August 4, 2010, from http://www.cs.utah.edu/~reinhard/cdrom/hdr.html

[34] *Recovering High Dynamic Range Radiance Maps from Photographs*, ICT.Debevec.org. Retrieved August 4, 2010, from http://ict.debevec.org/~debevec/Research/HDR/

[35] *High Dynamic Range Imaging and Tone-Mapping*, TU-Berlin.de. Retrieved April 14, 2010, from http://user.cs.tu-berlin.de/~eitz/hdr/

[36] *High Dynamic Range Rendering with iCAM* (page is no more available), CIS.RIT.edu. Retrieved July 6, 2010, from http://www.cis.rit.edu/mcsl/icam/hdr/

APPENDICES

APPENDIX A

MATLAB File for WDR Image Generation (makehdr.m)

(makehdr.m)

```
function hdr = makehdr(filenames, varargin)
%MAKEHDR    Create high dynamic range image.
%   HDR = MAKEHDR(FILES) creates the single-precision high dynamic
range
%    image HDR from the set of spatially registered low dynamic
range
%    images listed in the FILES cell array.  These files must
contain
%    EXIF exposure metadata.  The "middle" exposure value between
the
%   brightest and darkest images is used as the base exposure for
the
%   high dynamic range calculations.  (This value does not need to
%   appear in any particular file.)
%
%    HDR = MAKEHDR(FILES, PARAM1, VALUE1, ...) creates a high
dynamic range
%   image from the low dynamic range images in FILES, specifying
%    parameters and corresponding values that control various
aspects of
%    the image creation.  Parameter names can be abbreviated and
case does
%   not matter.
%
%   Parameters include:
%
%   'BaseFile'          Character array containing the name of the
file to
%                       use as the base exposure.
%
%    'ExposureValues'     A vector of exposure values, with one
element
%                       for each low dynamic range image in FILES.
An
%                            increase of one exposure value (EV)
corresponds
%                            to a doubling of exposure, while a
decrease in
%                            one EV corresponds to a halving of
exposure.
%                           Any positive value is allowed.  This
parameter
%                       overrides EXIF exposure metadata.
%
%   'RelativeExposure'  A vector of relative exposure values, with
one
%                         element for each low dynamic range image
in
%                        FILES.  An image with a relative exposure
(RE)
```

```
%                           of 0.5 has half as much exposure as an
image
%                           with an RE of 1.  An RE value of 3 has
three
%                         times the exposure of an image with an RE
of 1.
%                           This parameter overrides EXIF exposure
metadata.
%
%    'MinimumLimit'       A numeric scalar value in the range [0
255] that
%                         specifies the minimum "correctly exposed"
value.
%                         For each low dynamic range image, pixels
with
%                        smaller values are considered underexposed
and
%                           will not contribute to the final high
dynamic
%                        range image.
%
%    'MaximumLimit'       A numeric scalar value in the range [0
255] that
%                         specifies the maximum "correctly exposed"
value.
%                         For each low dynamic range image, pixels
with
%                         larger values are considered overexposed
and will
%                         not contribute to the final high dynamic
range
%                        image.
%
%   Note: Only one of the 'BaseFile', 'ExposureValues', and
%   'RelativeExposure' parameters may be used at a time.
%
%   Example
%   -------
%
%    Make a high dynamic range image from a series of six low
dynamic
%     range images that share the same f/stop number and have
different
%   exposure times.  Use TONEMAP to visualize the HDR image.
%
%        files = {'office_1.jpg', 'office_2.jpg', 'office_3.jpg',
...
%                 'office_4.jpg', 'office_5.jpg', 'office_6.jpg'};
%          expTimes = [0.0333, 0.1000, 0.3333, 0.6250, 1.3000,
4.0000];
%
%          hdr = makehdr(files, 'RelativeExposure', expTimes ./
expTimes(1));
%      rgb = tonemap(hdr);
%      figure; imshow(rgb)
%
%    Reference: Reinhard, et al. "High Dynamic Range Imaging."
2006. Ch. 4.
%
```

```
%   See also HDRREAD, HDRWRITE, TONEMAP.

%   Copyright 2007-2008 The MathWorks, Inc.
%   $Revision: 1.1.6.2 $  $Date: 2008/02/07 16:30:47 $

% Parse and check inputs.
iptcheckinput(filenames,    {'cell'},    {'nonempty'},    mfilename,
'files', 1);
options = parseArgs(varargin{:});
validateOptions(filenames, options);

% Get the minimum exposure image from the user or make a first
pass through
% the images to find the lowest exposure image.
if (~isempty(options.basefile))
    [baseTime, baseFStop] = getExposure(options.basefile);
elseif (isempty(options.relexp) && isempty(options.expvals))
    [baseTime, baseFStop] = getAverageExposure(filenames);
end

% Create output variables for an accumulator and the number of LDR
images
% that contributed to each pixel.
meta = imfinfo(filenames{1});

[hdr, properlyExposedCount] = makeContainers(meta);

someUnderExposed = false(size(hdr));
someOverExposed = false(size(hdr));
someProperlyExposed = false(size(hdr));

% Construct the HDR image by iterating over the LDR images.
for p = 1:numel(filenames)

    fname = filenames{p};

    if (~isempty(options.expvals))
        % Convert log2 EV equivalents to decimal values.
        relExposure = 2 .^ options.expvals(p);
    elseif (~isempty(options.relexp))
        relExposure = options.relexp(p);
    else
        [this_ExposureTime, this_FNumber] = getExposure(fname);
        relExposure = computeRelativeExposure(baseFStop, ...
                                              baseTime, ...
                                              this_FNumber, ...
                                              this_ExposureTime);
    end

    % Read the LDR image
    ldr = loadImage(fname, meta);

    underExposed = ldr < options.minclip;
    someUnderExposed = someUnderExposed | underExposed;

    overExposed = ldr > options.maxclip;
    someOverExposed = someOverExposed | overExposed;

    properlyExposed = ~(underExposed | overExposed);
```

```
    someProperlyExposed = someProperlyExposed | properlyExposed;

    properlyExposedCount(properlyExposed)                          =
properlyExposedCount(properlyExposed) + 1;

    % Remove over- and under-exposed values.
    ldr(~properlyExposed) = 0;

    % Bring the intensity of the LDR image into a common HDR
domain by
    % "normalizing" using the relative exposure, and then add it
to the
    % accumulator.
    hdr = hdr + single(ldr) ./ relExposure;

end

% Average the values in the accumulator by the number of LDR
images that
% contributed to each pixel to produce the HDR radiance map.
hdr = hdr ./ max(properlyExposedCount, 1);

% For pixels that were completely over-exposed, assign the maximum
% value computed for the properly exposed pixels.
hdr(someOverExposed & ~someUnderExposed & ~someProperlyExposed) =
max(hdr(someProperlyExposed));

% For pixels that were completely under-exposed, assign the
% minimum value computed for the properly exposed pixels.
hdr(someUnderExposed & ~someOverExposed & ~someProperlyExposed) =
min(hdr(someProperlyExposed));

% For pixels that were sometimes underexposed, sometimes
% overexposed, and never properly exposed, use roifill.
fillMask   =   imdilate(someUnderExposed   &   someOverExposed   &
~someProperlyExposed, ones(3,3));
if any(fillMask(:))
    hdr(:,:,1) = roifill(hdr(:,:,1), fillMask(:,:,1));
    hdr(:,:,2) = roifill(hdr(:,:,2), fillMask(:,:,2));
    hdr(:,:,3) = roifill(hdr(:,:,3), fillMask(:,:,3));
end

function [baseTime, baseFStop] = getExposure(filename)
%  Extract  the  exposure  values  from  a  file  containing  EXIF
metadata.

exif = getExposureDataFromFile(filename);
baseFStop = exif.FNumber;
baseTime = exif.ExposureTime;


function [baseTime, baseFStop] = getAverageExposure(filenames)
% Extract the average exposure (assuming constant illumination)
from a set
% of files containing EXIF metadata.  The average exposure may not
actually
% correspond to the exposure of any particular image.

minTime = 0;
```

```
minFStop = 0;
maxTime = 0;
maxFStop = 0;

% Look through all of the files and keep track of the least and
greatest
% exposure.
for p = 1:numel(filenames)

    exif = getExposureDataFromFile(filenames{p});

    if (p == 1)

        % First file.
        minFStop = exif.FNumber;
        minTime = exif.ExposureTime;
        maxFStop = exif.FNumber;
        maxTime = exif.ExposureTime;

    else

        % Nth file.
        if (computeRelativeExposure(minFStop, ...
                                    minTime, ...
                                    exif.FNumber, ...
                                    exif.ExposureTime) < 1)

            % Image has least exposure so far.
            minFStop = exif.FNumber;
            minTime = exif.ExposureTime;

        elseif (computeRelativeExposure(maxFStop, ...
                                        maxTime, ...
                                        exif.FNumber, ...
                                        exif.ExposureTime) > 1)

            % Image has most exposure so far.
            maxFStop = exif.FNumber;
            maxTime = exif.ExposureTime;

        end

    end

end

% Determine the "middle" exposure value. It's easier to
manipulate
% exposure time rather than f/stop.
re = computeRelativeExposure(minFStop, minTime, ...
                             maxFStop, maxTime);
baseFStop = minFStop;
baseTime  = minTime * log2(re);


function exif = getExposureDataFromFile(filename)
% Extract exposure metadata from a file containing EXIF.

try
```

```
    meta = imfinfo(filename);
    if isfield(meta, 'DigitalCamera')
        exif = meta.DigitalCamera;
    else

        error('Images:makehdr:exifFormat', ...
              'File %s does not have EXIF metadata.  %s', ...
              filename, ...
              'Use the ''ExposureValues'' or ''RelativeExposure''
parameter to provide exposure information.');

    end

catch ME

    if (isequal(ME.identifier, 'MATLAB:imfinfo:fileOpen'))

        error('Images:makehdr:fileNotFound', ...
              'File %s does not exist.', filename);

    else

        % Unexpected error.
        rethrow(ME)

    end
end

if (isempty(exif) || ...
    ~isstruct(exif) || ...
    ~isfield(exif, 'FNumber') || ...
    ~isfield(exif, 'ExposureTime'))

    error('Images:makehdr:noExposureMetadata', ...
          'File %s does not have exposure metadata.   Use the
''ExposureValues'' or ''RelativeExposure'' parameter to provide
exposure information.', filename)

end



function relExposure = computeRelativeExposure(f1, t1, f2, t2)

% Exposure varies directly with the exposure time and inversely
with the
% square of the F-stop number.
relExposure = (f1 / f2)^2 * (t2 / t1);


function options = parseArgs(varargin)
% Parse the parameter-value pairs, getting default values.

knownParams  = {'BaseFile',              'basefile',  '',  {'char'},
{'vector'};
                'ExposureValues',     'expvals',    [],  {'numeric'},
{'vector', 'real', 'finite', 'nonnan'};
                'RelativeExposure',  'relexp',     [],  {'numeric'},
```

```
{'vector', 'real', 'finite', 'positive', 'nonzero'};
                'MinimumLimit',       'minclip',    5, {'numeric'},
{'scalar', 'integer', 'real', 'nonnan', 'positive'};
                'MaximumLimit',       'maxclip', 250, {'numeric'},
{'scalar', 'integer', 'real', 'nonnan', 'positive'}};

options    =    parseParameterValuePairs(mfilename,    knownParams,
varargin{:});


function validateOptions(filenames, options)

% Make sure that mutually exclusive options aren't provided.
fieldCount = 0;

if (~isempty(options.basefile))
    fieldCount = fieldCount + 1;
end
if (~isempty(options.expvals))
    fieldCount = fieldCount + 1;
end
if (~isempty(options.relexp))
    fieldCount = fieldCount + 1;
end

if (fieldCount > 1)

    error('Images:makehdr:tooManyExposureParameters', ...
          'Only one of the following parameters is allowed:
''BaseFile'', ''ExposureValues'', or ''RelativeExposure''.')

end

% Make sure that the correct number of exposure-related values are
given.
if   (~isempty(options.expvals)   &&   (numel(options.expvals)   ~=
numel(filenames)))

    error('Images:makehdr:wrongExposureValuesCount', ...
          'The number of ''ExposureValues'' elements must match
the number of files.')

elseif   (~isempty(options.relexp)   &&   (numel(options.relexp)   ~=
numel(filenames)))

    error('Images:makehdr:wrongRelativeExposureCount', ...
          'The number of ''RelativeExposure'' elements must match
the number of files.')

end


function [hdr, counts] = makeContainers(meta)
% Create a floating point accumulator for the final HDR image and
a counter
% for the number of contributing images.

if (~isequal(meta.ColorType, 'truecolor'))
    error('Images:makehdr:notRGB', ...
```

```
            'Low dynamic range images must be RGB.')
end

hdr = zeros(meta.Height, meta.Width, 3, 'single');
counts = zeros(meta.Height, meta.Width, 3, 'single');


function ldr = loadImage(fname, meta)

ldr = imread(fname);

if (~isequal(size(ldr), [meta.Height, meta.Width, 3]))

    error('Images:makehdr:imageDimensions', ...
        'Low dynamic range image dimensions must agree.   File
''%s'' does not match.', ...
        fname);

end
```

APPENDIX B

MATLAB File For WDR Image Generation Using "makehdr"

(gen_matlab.m)

(gen_matlab.m)

```
% Load LDR images files***

% % Image / Scene: book
files = {'1-20.jpg','1-125.jpg','1-640.jpg'};

% % Image / Scene: lab1
% files = {'ldr1-50.jpg','ldr1-100.jpg','ldr1-250.jpg'};

% % Image / Scene: lab2
% files = {'Morning1_50.jpg','Morning1_250.jpg',...
%     'Morning1_700.jpg','Morning1_1000.jpg',...
%     'Morning1_2500.jpg','Morning1_10000.jpg'};

% % Image / Scene: window
% files = {'window_exp_15_1.jpg','window_exp_4_1.jpg',...
%     'window_exp_1_1.jpg','window_exp_1_4.jpg',...
%     'window_exp_1_15.jpg','window_exp_1_60.jpg',...
%     'window_exp_1_250.jpg','window_exp_1_1000.jpg',...
%     'window_exp_1_4000.jpg'};

% Specify their exposure time respectively***

% % Image / Scene: book
expTimes = [1/20 1/125 1/640];

% % Image / Scene: lab1
% expTimes = [1/50 1/100 1/250];

% % Image / Scene: lab2
% expTimes = [1/50 1/250 1/700 1/1000 1/2500 1/10000];

% % Image / Scene: window
% expTimes = [15 4 1 1/4 1/15 1/60 1/250 1/1000 1/4000];

% %***

% Calculate HDR image using base exposure obtained from middle
exposure
% between brightest and darkest images
tic;
wdr = makehdr(files, 'RelativeExposure', expTimes ./ expTimes(1));
time = toc;




% Write
```

```
hdrwrite(wdr, 'input book_matlab.hdr');
% hdrwrite(wdr, 'input lab1_matlab.hdr');
% hdrwrite(wdr, 'input lab2_matlab.hdr');
% hdrwrite(wdr, 'input window_matlab.hdr');
```

APPENDIX C

MATLAB File for WDR Image Generation Using P. Debevec And J. Malik Method (main_gen_pdjm.m)

(main_gen_pdjm.m)

```
% ------------------------------------------------------------------
---------
% Implements HDR generation
%
% Original author:
% Mathias Eitz
% m.eitz@tu-berlin.de
%
% Original implementation: January 2007
% Compatibility updates for new Matlab versions (7.8.0): April
2010
%
% 1.: Based on "Recovering High Dynamic Range Radiance Maps from
%      Photographs" by P. Debevec (SIGGRAPH'97). Some codes taken
from this
%     paper.
%
% 2.: Recovers a HDR radiance map from the range of LDR pictures
%
% 3.: Save the generated HDR radiance map as an HDR image file
using
%     "hdrwrite" function
%
% Modified by:
% Wazirah Md Esa
% ------------------------------------------------------------------
---------

% List LDR images with different exposure settings in descending
order***

% % Image / Scene: book
filenames = {'1-20.jpg','1-125.jpg','1-640.jpg'};

% % Image / Scene: lab1
% filenames = {'ldr1-50.jpg','ldr1-100.jpg','ldr1-250.jpg'};

% % Image / Scene: lab2
% filenames = {'Morning1_50.jpg','Morning1_250.jpg',...
%     'Morning1_700.jpg','Morning1_1000.jpg',...
%     'Morning1_2500.jpg','Morning1_10000.jpg'};

% % Image / Scene: window
% filenames = {'window_exp_15_1.jpg','window_exp_4_1.jpg',...
%     'window_exp_1_1.jpg','window_exp_1_4.jpg',...
%     'window_exp_1_15.jpg','window_exp_1_60.jpg',...
%     'window_exp_1_250.jpg','window_exp_1_1000.jpg',...
%     'window_exp_1_4000.jpg'};
```

```
% Specify their respective exposures in descending order***

% % Image / Scene: book
exposures = [1/20 1/125 1/640];

% % Image / Scene: lab1
% exposures = [1/50 1/100 1/250];

% % Image / Scene: lab2
% exposures = [1/50 1/250 1/700 1/1000 1/2500 1/10000];

% % Image / Scene: window
% exposures = [15 4 1 1/4 1/15 1/60 1/250 1/1000 1/4000];

% ***

% Load LDR images
tmp = imread(filenames{1});

numPixels = size(tmp,1) * size(tmp,2);
numExposures = size(filenames,2);

% Define smoothing factor, Lambda
l = 50;

% Compute weighting function value for each pixel
% weights = [];
weights = zeros(1,256);
for i=1:256
    weights(i) = weight(i,1,256);
end

% Load and sample the images
[zRed, zGreen, zBlue, sampleIndices] = makeImageMatrix(filenames,
numPixels);

% Create exposure matrix of B
B = zeros(size(zRed,1)*size(zRed,2), numExposures);
for i = 1:numExposures
    B(:,i) = log(exposures(i));
end

% Solve the system for each color channel
% Solve the response curve for each color channel
% Red channel
[gRed,lERed]=gsolve(zRed, B, l, weights);
% Green channel
[gGreen,lEGreen]=gsolve(zGreen, B, l, weights);
% Blue channel
[gBlue,lEBlue]=gsolve(zBlue, B, l, weights);

% save('gMatrix.mat','gRed', 'gGreen', 'gBlue');

% Compute and construct HDR radiance map
hdrMap = hdr(filenames, gRed, gGreen, gBlue, weights, B);

% ***
```

```
% Write HDR for Tone-Mapping using Photomatix
hdrwrite(hdrMap, 'input book_pdjm.hdr');
% hdrwrite(hdrMap, 'input lab1_pdjm.hdr');
% hdrwrite(hdrMap, 'input lab2_pdjm.hdr');
% hdrwrite(hdrMap, 'input window_pdjm2.hdr');
```

APPENDIX D

MATLAB File for WDR Image Generation Using P. Debevec And J. Malik Method (weight.m)

(weight.m)

```
function w = weight(z, zmin, zmax)
    if z <= 0.5 * (zmin + zmax)
        w = ((z - zmin) + 1);
    else
        w = ((zmax - z) + 1);
    end
```

APPENDIX E

MATLAB File for WDR Image Generation Using P. Debevec And J. Malik Method (makeimagematrix.m)

(makeImageMatrix.m)

```
% Takes relevant samples from the images for use in gsolve.m

function [ zRed, zGreen, zBlue, sampleIndices ] = makeImageMatrix(
filenames, numPixels )

    % Determine the number of differently exposed images
    numExposures = size(filenames,2);


    % Create the vector of sample indices
    % We need N(P-1) > (Zmax - Zmin)
    % Assuming the maximum (Zmax - Zmin) = 255,
    % N = (255 * 2) / (P-1) clearly fulfills this requirement
    numSamples = ceil(255*2 / (numExposures - 1)) * 2;

    % Create a random sampling matrix, telling us which
    % pixels of the original image we want to sample
    % using ceil fits the indices into the range [1,numPixels+1],
    % i.e. exactly the range of indices of zInput
    step = numPixels / numSamples;
    sampleIndices = floor((1:step:numPixels));
    sampleIndices = sampleIndices';

    % Allocate resulting matrices
    zRed = zeros(numSamples, numExposures);
    zGreen = zeros(numSamples, numExposures);
    zBlue = zeros(numSamples, numExposures);

    for i=1:numExposures

        % Read the nth image
        image = imread(filenames{i});

        % Sample the image for each color channel
        [zRedTemp,   zGreenTemp,   zBlueTemp]   =   sample(image,
sampleIndices);

        % Build the resulting, small image consisting
        % of samples of the original image
        zRed(:,i) = zRedTemp;
        zGreen(:,i) = zGreenTemp;
        zBlue(:,i) = zBlueTemp;
    end
```

APPENDIX F

MATLAB File for WDR Image Generation Using P. Debevec And J. Malik Method (sample.m)

(sample.m)

```
function [ red, green, blue ] = sample( image, sampleIndices )
    % Takes relevant samples of the input image

    redChannel = image(:,:,1);
    red = redChannel(sampleIndices);

    greenChannel = image(:,:,2);
    green = greenChannel(sampleIndices);

    blueChannel = image(:,:,3);
    blue = blueChannel(sampleIndices);
```

APPENDIX G

MATLAB File for WDR Image Generation Using P. Debevec And J. Malik Method (gsolve.m)

(gsolve.m)

```
% gsolve.m - Solve for imaging system response function
%
% Code taken from Paul Debevec's SIGGRAPH'97 paper "Recovering
% High Dynamic Range Radiance Maps from Photographs"
%
% Given a set of pixel values observed for several pixels in
several
% images with different exposure times, this function returns the
% imaging system's response function g as well as the log film
irradiance
% values for the observed pixels.
%
% Assumes:
%
% Zmin = 0
% Zmax = 255
%
% Arguments:
%
% Z(i,j) is the pixel values of pixel location number i in image j
% B(j) is the log delta t, or log shutter speed, for image j
% l is lamdba, the constant that determines the amount of
smoothness
% w(z) is the weighting function value for pixel value z
%
% Returns:
%
% g(z) is the log exposure corresponding to pixel value z
% lE(i) is the log film irradiance at pixel location i
%
function [g,lE]=gsolve(Z,B,l,w)
    n = 256;
    A = zeros(size(Z,1)*size(Z,2)+n+1,n+size(Z,1));
    b = zeros(size(A,1),1);

    %% Include the data-fitting equations
    k = 1;
    for i=1:size(Z,1)
        for j=1:size(Z,2)
            wij = w(Z(i,j)+1);
            A(k,Z(i,j)+1) = wij;
            A(k,n+i) = -wij;
            b(k,1) = wij * B(i,j);
            k=k+1;
        end
    end

    %% Fix the curve by setting its middle value to 0
```

```
    A(k,129) = 1;
    k=k+1;

    %% Include the smoothness equations
    for i=1:n-2
        A(k,i)=l*w(i+1); A(k,i+1)=-2*l*w(i+1); A(k,i+2)=l*w(i+1);
        k=k+1;
    end

    %% Solve the system using SVD
    x = A\b;
    g = x(1:n);
    lE = x(n+1:size(x,1));
```

APPENDIX H

MATLAB File for WDR Image Generation Using P. Debevec And J. Malik Method (hdr.m)

(hdr.m)

```
% Generates a hdr radiance map from a set of pictures
%
% Parameters:
%
%  filenames: a list of filenames containing the differently
exposed
% pictures used to make a hdr from
% gRed: camera response function for the red color channel
% gGreen: camera response function for the green color channel
% gBlue: camera response function for the blue color channel
function [ hdr ] = hdr( filenames, gRed, gGreen, gBlue, w, dt )

    numExposures = size(filenames,2);

    % Read the first image to get the width and height information
    image = imread(filenames{1});

    % Pre-allocate resulting HDR image
    hdr = zeros(size(image));
    sum = zeros(size(image));

    for i=1:numExposures

        % Adding pictures one by one
        image = double(imread(filenames{i}));

        wij = w(image + 1);
        sum = sum + wij;

        m(:,:,1) = (gRed(image(:,:,1) + 1) - dt(1,i));
        m(:,:,2) = (gGreen(image(:,:,2) + 1) - dt(1,i));
        m(:,:,3) = (gBlue(image(:,:,3) + 1) - dt(1,i));

        % If a pixel is saturated, its information and that
        % gathered from all prior pictures with longer
        % exposure times is unreliable. Thus we ignore its
        % influence on the weighted sum (influence of the
        % same pixel from prior pics with longer exposure time
        % ignored as well)

        saturatedPixels = ones(size(image));

        saturatedPixelsRed = find(image(:,:,1) == 255);
        saturatedPixelsGreen = find(image(:,:,2) == 255);
        saturatedPixelsBlue = find(image(:,:,3) == 255);

        % Mark the saturated pixels from a certain channel in
        % *all three* channels
```

```
        dim = size(image,1) * size(image,2);

        saturatedPixels(saturatedPixelsRed) = 0;
        saturatedPixels(saturatedPixelsRed + dim) = 0;
        saturatedPixels(saturatedPixelsRed + 2*dim) = 0;

        saturatedPixels(saturatedPixelsGreen) = 0;
        saturatedPixels(saturatedPixelsGreen + dim) = 0;
        saturatedPixels(saturatedPixelsGreen + 2*dim) = 0;

        saturatedPixels(saturatedPixelsBlue) = 0;
        saturatedPixels(saturatedPixelsBlue + dim) = 0;
        saturatedPixels(saturatedPixelsBlue + 2*dim) = 0;

        % Add the weighted sum of the current pic to the
        % resulting HDR radiance map
        hdr = hdr + (wij .* m);

        % Remove saturated pixels from the radiance map and
        % the sum (saturated pixels are zero in the
        % saturatedPixels matrix, all others are one)
        hdr = hdr .* saturatedPixels;
        sum = sum .* saturatedPixels;
    end

    % For those pixels that even in the picture with the
    % smallest exposure time still are saturated we approximate
    % the radiance only from that picture instead of taking
    % the weighted sum
    saturatedPixelIndices = find(hdr == 0);

    % Don't multiply with the weights since they are zero for
    % saturated pixels. m contains the logRadiance value from
    % the last pic, that one with the longest exposure time.
    hdr(saturatedPixelIndices) = m(saturatedPixelIndices);

    % Fix the sum for those pixels to avoid division by zero
    sum(saturatedPixelIndices) = 1;

    % Normalize
    hdr = hdr ./ sum;
    hdr = exp(hdr);
```

APPENDIX I

MATLAB File for Writing WDR Image File (hdrwrite.m)

(hdrwrite.m)

```
function hdrwrite(hdrImage, filename)
%HDRWRITE   Write Radiance .hdr file.
%     HDRWRITE(HDR, FILENAME) creates a Radiance .hdr file from
HDR, a
%     single- or double-precision high dynamic range RGB image.
The .hdr
%     file with the name FILENAME uses run-length encoding to
minimize file
%    size.
%
%    See also HDRREAD, MAKEHDR, TONEMAP.

%   Copyright 2007 The MathWorks, Inc.
%   $Revision: 1.1.6.1 $  $Date: 2007/11/09 20:22:56 $

iptcheckinput(hdrImage, {'single', 'double'}, ...
    {'finite', 'nonempty', 'nonnan', 'nonnegative', 'nonsparse',
'real'}, ...
    mfilename, 'HDR', 1);

iptcheckinput(filename, {'char'}, {'row'}, mfilename, 'FILENAME',
2);

% Convert the HDR RGB data to RBGE data.
rgbe = rgb2rgbe(permute(hdrImage, [2 1 3]));

% Write the RGBE data to the file.
fid = fopen(filename, 'w');
fprintf(fid, '#?RADIANCE\n');
fprintf(fid, '#Made with MATLAB\n');
fprintf(fid, 'FORMAT=32-bit_rle_rgbe\n');
fprintf(fid, '\n');
fprintf(fid, '-Y %d +X %d\n', size(hdrImage, 1), size(hdrImage,
2));

width = size(hdrImage, 2);

for row = 1:size(hdrImage,1)

    fwrite(fid, [2 2], 'uint8');
    fwrite(fid, width, 'uint16', 'ieee-be');

    for sample = 1:4

        dataStart = (row - 1) * width + 1;
        scanline = rleCoder(rgbe(dataStart:(dataStart + width -
1), sample), width);
        fwrite(fid, scanline, 'uint8');
    end
```

```
end
fclose(fid);
```

APPENDIX J

MATLAB File for Tone-Mapping (tonemap.m)

(tonemap.m)

```
function RGBldr = tonemap(RGBhdr, varargin)
%TONEMAP   Render high dynamic range image for viewing.
%    RGB = TONEMAP(HDR) performs tone mapping on the high dynamic
range
%    image HDR to a lower dynamic range image RGB suitable for
display.
%
%   RGB = TONEMAP(HDR, 'AdjustLightness', [LOW HIGH], ...) adjusts
the
%    overall lightness of the rendered image by passing the
luminance
%    values of the low dynamic range image to IMADJUST with the
values LOW
%   and HIGH, which are in the range [0, 1].
%
%   RGB = TONEMAP(HDR, 'AdjustSaturation', SCALE, ...) adjusts the
%    saturation of colors in the rendered image.  When SCALE is
greater
%   than 1, the colors are more saturated.  A SCALE value in the
range
%   [0, 1) results in less saturated colors.
%
%   RGB = TONEMAP(HDR, 'NumberOfTiles', [ROWS COLS], ...) sets the
number
%   of tiles used during the adaptive histogram equalization part
of the
%   tone mapping operation.  ROWS and COLS specify the number of
tile rows
%    and columns.  Both ROWS and COLS must be at least 2.  The
total number
%   of image tiles is equal to ROWS * COLS.  A larger number of
tiles
%   results in an image with greater local contrast.  The default
for ROWS
%   and COLS is 4.
%
%   Class Support
%   -------------
%   The high dynamic range image HDR must be a m-by-n-by-3 single
or
%    double array.  The output image RGB is an m-by-n-by-3 uint8
image.
%
%   Example
%   -------
%   Load a high dynamic range image, convert it to a low dynamic
range
%    image while deepening shadows and increasing saturation, and
display
%   the results.
```

```
%
%       hdr = hdrread('office.hdr');
%       imshow(hdr)
%       rgb = tonemap(hdr, 'AdjustLightness', [0.1 1], ...
%                     'AdjustSaturation', 1.5);
%       figure;
%       imshow(rgb);
%
%   See also ADAPTHISTEQ, HDRREAD, STRETCHLIM.

%   Copyright 2007-2009 The MathWorks, Inc.
%   $Revision: 1.1.6.4 $  $Date: 2009/11/09 16:25:33 $

% Parse and validate input arguments.
iptcheckinput(RGBhdr, {'single', 'double'}, {'real'}, mfilename,
'RGBhdr', 1);
if ((ndims(RGBhdr) ~= 3) || (size(RGBhdr, 3) ~= 3))
    error('Images:tonemap:badImageDimensions', ...
          'HDR image must be an m-by-n-by-3 single or double
array.')
end
options = parseArgs(varargin{:});

% Transform the HDR image to a new HDR image in the range [0,1] by
taking
% the base-2 logarithm and linearly scaling it.
[RGBlog2Scaled, hasNonzero] = lognormal(RGBhdr);

% Convert the image to a low dynamic range image by adaptive
histogram
% equalization.
if (hasNonzero)
    RGBldr = toneOperator(RGBlog2Scaled, ...
                          options.LRemap, ...
                          options.saturation, ...
                          options.numtiles);
else
    % "HDR" image only has zeros.  Return another image of zeros.
    RGBldr = RGBlog2Scaled;
end

RGBldr = uint8(RGBldr * 255);



function options = parseArgs(varargin)
% Get user-provided and default options.

% Create a structure with default values, and map actual param-
value pair
% names to convenient names for internal use.
knownParams  =  {'AdjustLightness',    'LRemap',           [0  1],
{'nonempty', 'vector', 'real', 'nonnan', 'positive'};
                'AdjustSaturation', 'saturation',     1, {'scalar',
'real', 'nonnan', 'positive'};
                'NumberOfTiles',          'numtiles',       [4  4],
{'nonempty',  'vector',  'integer',  'real',  'finite',  'positive',
'nonzero'}};
options = cell2struct(knownParams(:,3), knownParams(:,2), 1);
```

```
if (rem(nargin, 2) ~= 0)
    error('images:tonemap:paramValuePairs', ...
          'Named parameters must have a corresponding value.')
end

% Loop over the P-V pairs.
for p = 1:2:nargin
    % Get the parameter name.
    paramName = varargin{p};
    if (~ischar(paramName))
        error('images:tonemap:badParamName', ...
              'Parameter names be character arrays.')
    end

    % Look for the parameter amongst the possible values.
    idx = strmatch(lower(paramName), lower(knownParams(:,1)));
    if (isempty(idx))
        error('images:tonemap:unknownParamName', ...
              'Unknown parameter "%s".', paramName);
    elseif (numel(idx) > 1)
        error('images:tonemap:ambiguousParamName', ...
              'Ambiguous parameter "%s".', paramName);
    end

    % Validate the value.
    options.(knownParams{idx, 2}) = varargin{p+1};
    iptcheckinput(varargin{p+1}, ...
                  {'double'}, ...
                  knownParams{idx,4}, ...
                  mfilename, ...
                  knownParams{idx,1}, ...
                  p+2);  % p+2 = Param name + 1 + offset to first
arg
end



function [RGBlog2Scaled, hasNonzero] = lognormal(RGBhdr)
% Take the base-2 logarithm of an HDR image and return another HDR
in [0,1].

% Remove 0's from each channel.  This can change color quality,
but it's
% unlikely to have a big impact and prevents log(0) --> -inf.
That's worse.
minNonzero = min(RGBhdr(RGBhdr ~= 0));

if (isempty(minNonzero))

    RGBlog2Scaled = zeros(size(RGBhdr), class(RGBhdr));
    hasNonzero = false;

else

    RGBhdr(RGBhdr == 0) = minNonzero;

    % Ward's method equalizes the log-luminance histogram.
    RGBlog2 = log2(RGBhdr);
```

```
    RGBlog2Scaled = convertToImage(RGBlog2); % Normalize to [0,1]
    hasNonzero = true;

end



function RGBldr = toneOperator(RGBlog2Scaled, LRemap, saturation,
numtiles)
% Convert the image from HDR to LDR.

% Colorspaces for HDR imagery is tricky.  For simplicity, assign
the
% log-luminance image to be in sRGB.
Lab = sRGB2Lab(RGBlog2Scaled);

% Tone map the L* values from the RGB HDR to preserve overall
color as much
% as possible.  This decreases global saturation, which can be
reintroduced
% by scaling the a* and b* channels.
Lab(:,:,1) = Lab(:,:,1) ./ 100;
Lab(:,:,1) = adapthisteq(Lab(:,:,1), 'NumTiles', numtiles);
Lab(:,:,1) = imadjust(Lab(:,:,1), LRemap, [0 1]) * 100;
Lab(:,:,2) = Lab(:,:,2) * saturation;
Lab(:,:,3) = Lab(:,:,3) * saturation;

% Convert the image back to sRGB.
RGBldr = Lab2sRGB(Lab);



function y = convertToImage(x)
% Rescale an image with arbitrary range to [0,1].

xMin = min(x(:));
xMax = max(x(:));

if (xMin == xMax)

    % Avoid dividing by zero.
    if (xMin == 0)
        y = x;
    else
        y = x ./ xMin;
    end

else
    % Linearly scale the values to [0,1].
    y = (x - xMin) ./ (xMax - xMin);
end



function Lab = sRGB2Lab(rgb)
% Convert sRGB values in the range [0,1] to Lab via XYZ.

dims = size(rgb);
rgb = reshape(permute(rgb, [3 1 2]), [3, dims(1) * dims(2)]);
```

```
Lab = XYZ2Lab(sRGB2XYZ(rgb));
Lab = permute(reshape(Lab, [3, dims(1), dims(2)]), [2 3 1]);



function rgb = Lab2sRGB(lab)
% Convert Lab values to sRGB values in the range [0,1] via XYZ.

dims = size(lab);
lab = reshape(permute(lab, [3 1 2]), [3, dims(1) * dims(2)]);
rgb = XYZ2sRGB(Lab2XYZ(lab));
rgb = permute(reshape(rgb, [3, dims(1), dims(2)]), [2 3 1]);


function xyz = sRGB2XYZ(rgb)
% Convert sRGB values to XYZ assuming a D65 whitepoint.

%  See  <http://brucelindbloom.com/index.html?Eqn_RGB_to_XYZ.html>
for
% implementation details.
M = [0.412424    0.212656    0.0193324
     0.357579    0.715158    0.119193
     0.180464    0.0721856   0.950444];

mask = (rgb > 0.04045);

rgb(mask) = ((rgb(mask) + 0.055) ./ 1.055) .^ 2.4;
rgb(~mask) = rgb(~mask) ./ 12.92;

xyz = M' * rgb;


function lab = XYZ2Lab(xyz)
% Convert XYZ to Lab using a D65 whitepoint.

%  See  <http://brucelindbloom.com/index.html?Eqn_XYZ_to_Lab.html>
for
% implementation details and explanation of E and K.
E = 216/24389;
K = 24389/27;

% Whitepoint adjustment.
xyz(1,:) = xyz(1,:) ./ 0.95047;
xyz(3,:) = xyz(3,:) ./ 1.08883;

mask = (xyz > E);

f = xyz;
f(mask) = f(mask) .^ (1/3);
f(~mask) = (K * f(~mask) + 16) ./ 116;

lab = [116 * f(2,:) - 16;
       500 * (f(1,:) - f(2,:));
       200 * (f(2,:) - f(3,:))];



function xyz = Lab2XYZ(lab)
% Convert Lab to XYZ using a D65 whitepoint.
```

```
%  See  <http://brucelindbloom.com/index.html?Eqn_Lab_to_XYZ.html>
for
% implementation details and explanation of E and K.
E = 216/24389;
K = 24389/27;

f = lab;
xyz = f;

mask = lab(1,:) > K*E;
xyz(2,mask) = ((lab(1,mask) + 16) ./ 116) .^ 3;
xyz(2,~mask) = lab(1,~mask) ./ K;

mask = (xyz(2,:) > E);
f(2,mask) = (lab(1,mask) + 16) ./ 116;
f(2,~mask) = (K * xyz(2,~mask) + 16) ./ 116;

f(1,:) = lab(2,:) ./ 500 + f(2,:);
f(3,:) = f(2,:) - lab(3,:) ./ 200;

tmp = f(1,:) .^ 3;
mask = (tmp > E);
xyz(1,mask) = tmp(mask);
xyz(1,~mask) = (116 * f(1,~mask) - 16) ./ K;

tmp = f(3,:) .^ 3;
mask = (tmp > E);
xyz(3,mask) = tmp(mask);
xyz(3,~mask) = (116 * f(3,~mask) - 16) ./ K;

% Whitepoint adjustment.
xyz(1,:) = xyz(1,:) * 0.95047;
xyz(3,:) = xyz(3,:) * 1.08883;



function rgb = XYZ2sRGB(xyz)
% Convert XYZ to sRGB using a D65 whitepoint.

%  See  <http://brucelindbloom.com/index.html?Eqn_XYZ_to_RGB.html>
for
% implementation details.
M = [0.412424    0.212656    0.0193324
     0.357579    0.715158    0.119193
     0.180464    0.0721856   0.950444];

rgb = M' \ xyz;

mask = (rgb > 0.0031308);
rgb(mask) = ((1.055 * rgb(mask)) .^ (1 / 2.4)) - 0.055;
rgb(~mask) = 12.92 * rgb(~mask);
```

APPENDIX K

MATLAB File for Tone-Mapping Using "tonemap" (matlabtonemap.m)

(matlabtonemap.m)

```
%% MATLAB Tone-Mapping using tonemap function

% close all
% clear all
% clc

% start timer
tic;

% load and read input image
% img = hdrread('input bristolb.hdr');
% img = hdrread('input memorial.hdr');
% img = hdrread('input rosette.hdr');

% img = hdrread('input book_matlab.hdr');
% img = hdrread('input lab1_matlab.hdr');
% img = hdrread('input lab2_matlab.hdr');
% img = hdrread('input window_matlab.hdr');

img = hdrread('input book_pdjm.hdr');
% img = hdrread('input lab1_pdjm.hdr');
% img = hdrread('input lab2_pdjm.hdr');
% img = hdrread('input window_pdjm.hdr');

% img = hdrread('input book_photomatix.hdr');
% img = hdrread('input lab1_photomatix.hdr');
% img = hdrread('input lab2_photomatix.hdr');
% img = hdrread('input window_photomatix.hdr');

hdr = double(img);

% tone-mapping
NewImg = tonemap(hdr);

% end timer
time = toc;

% display
imshow(NewImg);

% % write
% imwrite(NewImg, 'output bristolb local_reinhard.jpg');
% imwrite(NewImg, 'output memorial local_reinhard.jpg');
% imwrite(NewImg, 'output rosette local_reinhard.jpg');

imwrite(NewImg, 'output book_matlab local_reinhard.jpg');
% imwrite(NewImg, 'output lab1_matlab local_reinhard.jpg');
% imwrite(NewImg, 'output lab2_matlab local_reinhard.jpg');
```

```
% imwrite(NewImg, 'output window_matlab local_reinhard.jpg');

% imwrite(NewImg, 'output book_pdjm local_reinhard.jpg');
% imwrite(NewImg, 'output lab1_pdjm local_reinhard.jpg');
% imwrite(NewImg, 'output lab2_pdjm local_reinhard.jpg');
% imwrite(NewImg, 'output window_pdjm local_reinhard.jpg');

% imwrite(ldrlocal, 'output book_photomatix local_reinhard.jpg');
% imwrite(ldrlocal, 'output lab1_photomatix local_reinhard.jpg');
% imwrite(ldrlocal, 'output lab2_photomatix local_reinhard.jpg');
% imwrite(ldrlocal, 'output window_photomatix
local_reinhard.jpg');
```

APPENDIX L

MATLAB File for Logarithmic Tone-Mapping (logtonemap.m)

(logtonemap.m)

```
%% Logarithmic Tone-Mapping

close all
clear all
clc

% start timer
tic;

% load and read input image
% img = hdrread('input bristolb.hdr');
% img = hdrread('input memorial.hdr');
% img = hdrread('input rosette.hdr');

img = hdrread('input book_matlab.hdr');
% img = hdrread('input lab1_matlab.hdr');
% img = hdrread('input lab2_matlab.hdr');
% img = hdrread('input window_matlab.hdr');

% img = hdrread('input book_pdjm.hdr');
% img = hdrread('input lab1_pdjm.hdr');
% img = hdrread('input lab2_pdjm.hdr');
% img = hdrread('input window_pdjm.hdr');

% img = hdrread('input book_photomatix.hdr');
% img = hdrread('input lab1_photomatix.hdr');
% img = hdrread('input lab2_photomatix.hdr');
% img = hdrread('input window_photomatix.hdr');

hdr = double(img);

% luminance RGB = 0.270 R + 0.670 G + 0.060 B
Lum = 0.270 * hdr(:,:,1) + 0.670 * hdr(:,:,2) + 0.060 *
hdr(:,:,3);

% the maximum luminance
Lmax = max(max(Lum));

% luminance of display
Lumd = log10(1+Lum) ./  log10(1+Lmax);

% s: saturation values, between 0 and 1
% 1 keeps color ratios constant
% smaller values, image will appear more desaturated
s = 0.6;

% red channel
RedChannel = hdr(:,:,1);
NewRedChannel = Lumd .* ((RedChannel ./ Lum).^s);

% green channel
```

```
GreenChannel = hdr(:,:,2);
NewGreenChannel = Lumd .* ((GreenChannel ./ Lum).^s);

% blue channel
BlueChannel = hdr(:,:,3);
NewBlueChannel = Lumd .* ((BlueChannel ./ Lum).^s);

% recombine luminance values into a color image
NewImg(:,:,1) = NewRedChannel;
NewImg(:,:,2) = NewGreenChannel;
NewImg(:,:,3) = NewBlueChannel;

% end timer
time = toc;

% display
imshow(NewImg);

% write
% imwrite(NewImg, 'output bristolb logtonemap.jpg');
% imwrite(NewImg, 'output memorial logtonemap.jpg');
% imwrite(NewImg, 'output rosette logtonemap.jpg');

imwrite(NewImg, 'output book_matlab logtonemap.jpg');
% imwrite(NewImg, 'output lab1_matlab logtonemap.jpg');
% imwrite(NewImg, 'output lab2_matlab logtonemap.jpg');
% imwrite(NewImg, 'output window_matlab logtonemap.jpg');

% imwrite(NewImg, 'output book_pdjm logtonemap.jpg');
% imwrite(NewImg, 'output lab1_pdjm logtonemap.jpg');
% imwrite(NewImg, 'output lab2_pdjm logtonemap.jpg');
% imwrite(NewImg, 'output window_pdjm logtonemap.jpg');

% imwrite(NewImg, 'output book_photomatix logtonemap.jpg');
% imwrite(NewImg, 'output lab1_photomatix logtonemap.jpg');
% imwrite(NewImg, 'output lab2_photomatix logtonemap.jpg');
% imwrite(NewImg, 'output window_photomatix logtonemap.jpg');
```

APPENDIX M

MATLAB File for Modified Logarithmic Tone-Mapping (logtonemap_mod.m)

(logtonemap_mod.m)

```
%% Modified Logarithmic Tone-Mapping

% close all
% clear all
% clc

% start timer
tic;

% load and read input image
% img = hdrread('input bristolb.hdr');
% img = hdrread('input memorial.hdr');
% img = hdrread('input rosette.hdr');

img = hdrread('input book_matlab.hdr');
% img = hdrread('input lab1_matlab.hdr');
% img = hdrread('input lab2_matlab.hdr');
% img = hdrread('input window_matlab.hdr');

% img = hdrread('input book_pdjm.hdr');
% img = hdrread('input lab1_pdjm.hdr');
% img = hdrread('input lab2_pdjm.hdr');
% img = hdrread('input window_pdjm.hdr');

% img = hdrread('input book_photomatix.hdr');
% img = hdrread('input lab1_photomatix.hdr');
% img = hdrread('input lab2_photomatix.hdr');
% img = hdrread('input window_photomatix.hdr');

hdr = double(img);

% luminance RGB = 0.270 R + 0.670 G + 0.060 B
%  Lum  =  0.270  *  hdr(:,:,1)  +  0.670  *  hdr(:,:,2)  +  0.060  *
hdr(:,:,3);
LumR = 0.270 * hdr(:,:,1);
LumG = 0.670 * hdr(:,:,2);
LumB = 0.060 * hdr(:,:,3);

% LumR = hdr(:,:,1);
% LumG = hdr(:,:,2);
% LumB = hdr(:,:,3);

% the maximum luminance
LmaxR = max(max(LumR));
LmaxG = max(max(LumG));
LmaxB = max(max(LumB));

% luminance of display
```

```
LumdR = log10(1+LumR) ./  log10(1+LmaxR);
LumdG = log10(1+LumG) ./  log10(1+LmaxG);
LumdB = log10(1+LumB) ./  log10(1+LmaxB);

% s: saturation values, between 0 and 1
% 1 keeps color ratios constant
% smaller values, image will appear more desaturated
s = 0.6;

% red channel
RedChannel = hdr(:,:,1);
NewRedChannel = LumdR .* ((RedChannel ./ LumR).^s);

% green channel
GreenChannel = hdr(:,:,2);
NewGreenChannel = LumdG .* ((GreenChannel ./ LumG).^s);

% blue channel
BlueChannel = hdr(:,:,3);
NewBlueChannel = LumdB .* ((BlueChannel ./ LumB).^s);

% recombine luminance values into a color image
NewImg(:,:,1) = NewRedChannel;
NewImg(:,:,2) = NewGreenChannel;
NewImg(:,:,3) = NewBlueChannel;

% end timer
time = toc;

% display
imshow(NewImg)

% write
% imwrite(NewImg, 'output bristolb logtonemap_mod.jpg');
% imwrite(NewImg, 'output memorial logtonemap_mod.jpg');
% imwrite(NewImg, 'output rosette logtonemap_mod.jpg');

imwrite(NewImg, 'output book_matlab logtonemap_mod.jpg');
% imwrite(NewImg, 'output lab1_matlab logtonemap_mod.jpg');
% imwrite(NewImg, 'output lab2_matlab logtonemap_mod.jpg');
% imwrite(NewImg, 'output window_matlab logtonemap_mod.jpg');

% imwrite(NewImg, 'output book_pdjm logtonemap_mod.jpg');
% imwrite(NewImg, 'output lab1_pdjm logtonemap_mod.jpg');
% imwrite(NewImg, 'output lab2_pdjm logtonemap_mod.jpg');
% imwrite(NewImg, 'output window_pdjm logtonemap_mod.jpg');

% imwrite(NewImg, 'output book_photomatix logtonemap_mod.jpg');
% imwrite(NewImg, 'output lab1_photomatix logtonemap_mod.jpg');
% imwrite(NewImg, 'output lab2_photomatix logtonemap_mod.jpg');
% imwrite(NewImg, 'output window_photomatix logtonemap_mod.jpg');
```

APPENDIX N

MATLAB File for Exponential Tone-Mapping (exptonemap.m)

(exptonemap.m)

```
%% Exponential Tone-Mapping

close all
clear all
clc

% start timer
tic;

% load and read input image
% img = hdrread('input bristolb.hdr');
% img = hdrread('input memorial.hdr');
% img = hdrread('input rosette.hdr');

img = hdrread('input book_matlab.hdr');
% img = hdrread('input lab1_matlab.hdr');
% img = hdrread('input lab2_matlab.hdr');
% img = hdrread('input window_matlab.hdr');

% img = hdrread('input book_pdjm.hdr');
% img = hdrread('input lab1_pdjm.hdr');
% img = hdrread('input lab2_pdjm.hdr');
% img = hdrread('input window_pdjm.hdr');

% img = hdrread('input book_photomatix.hdr');
% img = hdrread('input lab1_photomatix.hdr');
% img = hdrread('input lab2_photomatix.hdr');
% img = hdrread('input window_photomatix.hdr');

hdr = double(img);

% luminance RGB = 0.270 R + 0.670 G + 0.060 B
Lum = 0.270 * hdr(:,:,1) + 0.670 * hdr(:,:,2) + 0.060 *
hdr(:,:,3);

% the number of pixels
numpixels = size(Lum,1) * size(Lum,2);

% average luminance
% delta is a small value to avoid singularity that occurs if black
pixels
% are present in the image
delta = 0.0001; %0.0001, 0.001, 0.01, 0.1, 1.0
Lumav = exp((1/numpixels)*(sum(sum(log(delta+Lum)))));

% luminance of display
Lumd = 1-exp(-(Lum./Lumav));

% s: saturation values, between 0 and 1
% 1 keeps color ratios constant
```

```
% smaller values, image will appear more desaturated
s = 0.6;

% red channel
RedChannel = hdr(:,:,1);
NewRedChannel = Lumd .* ((RedChannel ./ Lum).^s);

% green channel
GreenChannel = hdr(:,:,2);
NewGreenChannel = Lumd .* ((GreenChannel ./ Lum).^s);

% blue channel
BlueChannel = hdr(:,:,3);
NewBlueChannel = Lumd .* ((BlueChannel ./ Lum).^s);

% recombine luminance values into a color image
NewImg(:,:,1) = NewRedChannel;
NewImg(:,:,2) = NewGreenChannel;
NewImg(:,:,3) = NewBlueChannel;

% end timer
time = toc;

% display
imshow(NewImg);

% write
% imwrite(NewImg, 'output bristolb exptonemap.jpg');
% imwrite(NewImg, 'output memorial exptonemap.jpg');
% imwrite(NewImg, 'output rosette exptonemap.jpg');

imwrite(NewImg, 'output book_matlab exptonemap.jpg');
% imwrite(NewImg, 'output lab1_matlab exptonemap.jpg');
% imwrite(NewImg, 'output lab2_matlab exptonemap.jpg');
% imwrite(NewImg, 'output window_matlab exptonemap.jpg');

% imwrite(NewImg, 'output book_pdjm exptonemap.jpg');
% imwrite(NewImg, 'output lab1_pdjm exptonemap.jpg');
% imwrite(NewImg, 'output lab2_pdjm exptonemap.jpg');
% imwrite(NewImg, 'output window_pdjm exptonemap.jpg');

% imwrite(NewImg, 'output book_photomatix exptonemap.jpg');
% imwrite(NewImg, 'output lab1_photomatix exptonemap.jpg');
% imwrite(NewImg, 'output lab2_photomatix exptonemap.jpg');
% imwrite(NewImg, 'output window_photomatix exptonemap.jpg');
```

APPENDIX O

MATLAB File for Modified Exponential Tone-Mapping

(exptonemap_mod.m)

(exptonemap_mod.m)

```
%% Modified Exponential Tone-Mapping

% close all
% clear all
% clc

% start timer
tic;

% load and read input image
% img = hdrread('input bristolb.hdr');
% img = hdrread('input memorial.hdr');
% img = hdrread('input rosette.hdr');

img = hdrread('input book_matlab.hdr');
% img = hdrread('input lab1_matlab.hdr');
% img = hdrread('input lab2_matlab.hdr');
% img = hdrread('input window_matlab.hdr');

% img = hdrread('input book_pdjm.hdr');
% img = hdrread('input lab1_pdjm.hdr');
% img = hdrread('input lab2_pdjm.hdr');
% img = hdrread('input window_pdjm.hdr');

% img = hdrread('input book_photomatix.hdr');
% img = hdrread('input lab1_photomatix.hdr');
% img = hdrread('input lab2_photomatix.hdr');
% img = hdrread('input window_photomatix.hdr');

hdr = double(img);

% luminance RGB = 0.270 R + 0.670 G + 0.060 B
%  Lum  =  0.270  *  hdr(:,:,1)  +  0.670  *  hdr(:,:,2)  +  0.060  *
hdr(:,:,3);
LumR = 0.270 * hdr(:,:,1);
LumG = 0.670 * hdr(:,:,2);
LumB = 0.060 * hdr(:,:,3);

% the number of pixels
numpixels = size(LumR,1) * size(LumR,2);

% average luminance
% delta is a small value to avoid singularity that occurs if black
pixels
% are present in the image
delta = 0.0001; %0.0001, 0.001, 0.01, 0.1, 1.0
LumavR = exp((1/numpixels)*(sum(sum(log(delta+LumR)))));
LumavG = exp((1/numpixels)*(sum(sum(log(delta+LumG)))));
```

```
LumavB = exp((1/numpixels)*(sum(sum(log(delta+LumB)))));

% luminance of display
LumdR = 1-exp(-(LumR./LumavR));
LumdG = 1-exp(-(LumG./LumavG));
LumdB = 1-exp(-(LumB./LumavB));

% s: saturation values, between 0 and 1
% 1 keeps color ratios constant
% smaller values, image will appear more desaturated
s = 0.6;

% red channel
RedChannel = hdr(:,:,1);
NewRedChannel = LumdR .* ((RedChannel ./ LumR).^s);

% green channel
GreenChannel = hdr(:,:,2);
NewGreenChannel = LumdG .* ((GreenChannel ./ LumG).^s);

% blue channel
BlueChannel = hdr(:,:,3);
NewBlueChannel = LumdB .* ((BlueChannel ./ LumB).^s);

% recombine luminance values into a color image
NewImg(:,:,1) = NewRedChannel;
NewImg(:,:,2) = NewGreenChannel;
NewImg(:,:,3) = NewBlueChannel;

% end timer
time = toc;

% display
imshow(NewImg);

% write
% imwrite(NewImg, 'output bristolb exptonemap_mod.jpg');
% imwrite(NewImg, 'output memorial exptonemap_mod.jpg');
% imwrite(NewImg, 'output rosette exptonemap_mod.jpg');

imwrite(NewImg, 'output book_matlab exptonemap_mod.jpg');
% imwrite(NewImg, 'output lab1_matlab exptonemap_mod.jpg');
% imwrite(NewImg, 'output lab2_matlab exptonemap_mod.jpg');
% imwrite(NewImg, 'output window_matlab exptonemap_mod.jpg');

% imwrite(NewImg, 'output book_pdjm exptonemap_mod.jpg');
% imwrite(NewImg, 'output lab1_pdjm exptonemap_mod.jpg');
% imwrite(NewImg, 'output lab2_pdjm exptonemap_mod.jpg');
% imwrite(NewImg, 'output window_pdjm exptonemap_mod.jpg');

% imwrite(NewImg, 'output book_photomatix exptonemap_mod.jpg');
% imwrite(NewImg, 'output lab1_photomatix exptonemap_mod.jpg');
% imwrite(NewImg, 'output lab2_photomatix exptonemap_mod.jpg');
% imwrite(NewImg, 'output window_photomatix exptonemap_mod.jpg');
```

APPENDIX P

MATLAB File for Reinhard et al Global Tone-Mapping (reinhard_global.m)

(reinhard_global.m)

```
%% Reinhard et al Global Tone-Mapping
%    Original implementation was done by Mathiaz Ethiz, from
Technische
%  Universität, Berlin (TU Berlin) in January 2007

% Modified by:
% Wazirah Md Esa

% start timer
tic;

% load and read input image
% img = hdrread('input bristolb.hdr');
% img = hdrread('input memorial.hdr');
% img = hdrread('input rosette.hdr');

img = hdrread('input book_matlab.hdr');
% img = hdrread('input lab1_matlab.hdr');
% img = hdrread('input lab2_matlab.hdr');
% img = hdrread('input window_matlab.hdr');

% img = hdrread('input book_pdjm.hdr');
% img = hdrread('input lab1_pdjm.hdr');
% img = hdrread('input lab2_pdjm.hdr');
% img = hdrread('input window_pdjm.hdr');

% img = hdrread('input book_photomatix.hdr');
% img = hdrread('input lab1_photomatix.hdr');
% img = hdrread('input lab2_photomatix.hdr');
% img = hdrread('input window_photomatix.hdr');

hdr = double(img);

% original implementation, luminancemap = 0.2125 R + 0.7154 G +
0.0721 B
% luminancemap = 0.2125 * hdr(:,:,1) + 0.7154 * hdr(:,:,2) +
0.0721 * hdr(:,:,3);

% other implementation, luminancemap = 0.270 R + 0.670 G + 0.060 B
luminancemap = 0.270 * hdr(:,:,1) + 0.670 * hdr(:,:,2) + 0.060 *
hdr(:,:,3);

% a: a parameter related to key of the scene
% range from 0.045, 0.09, 0.18, 0.36, and 0.72
a = 0.72;

% assign small delta value to avoid taking log(0) when
encountering black
% pixels in the luminance map
delta = 0.0001;
```

```
% saturation: saturation values, between 0 and 1
% 1 keeps color ratios constant
% smaller values, image will appear more desaturated
saturation = 1.0;

numpixels = size(hdr,1) * size(hdr,2);

% compute the key of the scene, a measure of the average
logarithmic
% luminance
% i.e. the subjective brightness of the image a human would
approximately
% perceive
key = exp((1/numpixels)*(sum(sum(log(delta + luminancemap)))));

% scale to desired brightness level as defined by the user
scaledluminance = luminancemap * (a/key);

% display luminance, without Lwhite
% luminanceglobal = scaledluminance ./ (scaledluminance + 1);

% display luminance, with Lwhite
% Lwhite default = max(max(scaledluminance));
% if Lwhite < 1, give subtle contrast enhancement
Lwhite = max(max(scaledluminance));
luminancedisplay = (scaledluminance .* (1 + (scaledluminance ./
(Lwhite .^2)))) ./ (scaledluminance + 1);

% re-apply color according to Fattals paper "Gradient Domain High
Dynamic
% Range Compression"
ldrglobal = zeros(size(hdr));
for i = 1:3

    % (hdr(:,:,i) ./ luminance) must be between 0 and 1
    % but HDR image often contains bigger values than luminance
    % so resulting ldr image needs to be clamped

    ldrglobal(:,:,i) = ((hdr(:,:,i) ./ luminancemap) .^
saturation) .* luminancedisplay;

end

% clamp resulting LDR image to 1
indices = find(ldrglobal > 1);
ldrglobal(indices) = 1;

% end timer
time = toc;

% display
imshow(ldrglobal);

% write
% imwrite(ldrglobal, 'output bristolb global_reinhard.jpg');
% imwrite(ldrglobal, 'output memorial global_reinhard.jpg');
% imwrite(ldrglobal, 'output rosette global_reinhard.jpg');
```

```
imwrite(ldrglobal, 'output book_matlab global_reinhard.jpg');
% imwrite(ldrglobal, 'output lab1_matlab global_reinhard.jpg');
% imwrite(ldrglobal, 'output lab2_matlab global_reinhard.jpg');
% imwrite(ldrglobal, 'output window_matlab global_reinhard.jpg');

% imwrite(ldrglobal, 'output book_pdjm global_reinhard.jpg');
% imwrite(ldrglobal, 'output lab1_pdjm global_reinhard.jpg');
% imwrite(ldrglobal, 'output lab2_pdjm global_reinhard.jpg');
% imwrite(ldrglobal, 'output window_pdjm global_reinhard.jpg');

%         imwrite(ldrglobal,         'output         book_photomatix
global_reinhard.jpg');
%         imwrite(ldrglobal,         'output         lab1_photomatix
global_reinhard.jpg');
%         imwrite(ldrglobal,         'output         lab2_photomatix
global_reinhard.jpg');
%        imwrite(ldrglobal,        'output        window_photomatix
global_reinhard.jpg');
```

APPENDIX Q

MATLAB File for Reinhard et al Local Tone-Mapping (reinhard_local.m)

(reinhard_local.m)

```
%% Reinhard et al Local Tone-Mapping
%    Original implementation was done by Mathiaz Ethiz, from
Technische
%  Universität, Berlin (TU Berlin) in January 2007

% Modified by:
% Wazirah Md Esa

% start timer
tic;

% load and read input image
% img = hdrread('input bristolb.hdr');
% img = hdrread('input memorial.hdr');
% img = hdrread('input rosette.hdr');

img = hdrread('input book_matlab.hdr');
% img = hdrread('input lab1_matlab.hdr');
% img = hdrread('input lab2_matlab.hdr');
% img = hdrread('input window_matlab.hdr');

% img = hdrread('input book_pdjm.hdr');
% img = hdrread('input lab1_pdjm.hdr');
% img = hdrread('input lab2_pdjm.hdr');
% img = hdrread('input window_pdjm.hdr');

% img = hdrread('input book_photomatix.hdr');
% img = hdrread('input lab1_photomatix.hdr');
% img = hdrread('input lab2_photomatix.hdr');
% img = hdrread('input window_photomatix.hdr');

hdr = double(img);

% original implementation, luminancemap = 0.2125 R + 0.7154 G +
0.0721 B
% luminancemap = 0.2125 * hdr(:,:,1) + 0.7154 * hdr(:,:,2) +
0.0721 *
% hdr(:,:,3);

% other implementation, luminancemap = 0.270 R + 0.670 G + 0.060 B
luminancemap = 0.270 * hdr(:,:,1) + 0.670 * hdr(:,:,2) + 0.060 *
hdr(:,:,3);

% key: range from 0.045, 0.09, ,0.18, 0.36, and 0.72
key = 0.72;

% saturation: saturation values, between 0 and 1
% 1 keeps color ratios constant
% smaller values, image will appear more desaturated
saturation = 0.6;
```

```
alpha = 1 / (2*sqrt(2));
phi = 8;
eps = 0.05;

v = zeros(size(luminancemap,1), size(luminancemap,2), 8);
v1 = zeros(size(luminancemap,1), size(luminancemap,2), 8);

% compute nine gaussian filtered version of the hdr luminance map,
such
% that we can compute eight difference.
% each image gets filtered by a standard gaussian filter, each
time with
% sigma 1.6 times higher than the sigma of the predecessor.

for scale = 1:(8+1)
    s = 1.6 ^ (scale-1);
    sigma = alpha * s;

    % discretize gaussian filter to a fixed kernel size
    % a radius of 2*sigma should keep the error low enough

    kernelradius = ceil(2*sigma);
    kernelsize = 2*kernelradius+1;

    gausskernelhori = fspecial('gaussian', [kernelsize 1], sigma);
    v1(:,:,scale) = conv2(luminancemap, gausskernelhori, 'same');

    gausskernelvert = fspecial('gaussian', [1 kernelsize], sigma);
    v1(:,:,scale) = conv2(v1(:,:,scale), gausskernelvert, 'same');
end

for i = 1:8
    v(:,:,i) = abs((v1(:,:,i)) - v1(:,:,i+1)) ./ ((2^phi)*key /
(s^2) + v1(:,:,i));
end

sm = zeros(size(v,1), size(v,2));

for i = 1:size(v,1)
    for j = 1:size(v,2)
        for scale = 1:size(v,3)

            % choose the biggest possible neighborhood where
v(i,j,scale)
            % is still smaller than a certain epsilon.
            % note that we need to choose that neighborhood which
is as big
            % as possible but all smaller neighborhoods also
fulfill
            % v(i,j,scale) < eps

            if v(i,j,scale) > eps

                % if we have a high contrast change in the first
scale, we
                % can only use that one

                if (scale == 1)
```

```
                    sm(i,j) = 1;
                end

                % if we have a contrast change bigger than
epsilon, we know
                % that in scale scale-1 the contrast change was
smaller
                % than epsilon, we use that one

                if (scale > 1)
                    sm(i,j) = scale-1;
                end

                break;
            end
        end
    end
end

% all areas in the image that have very small variations and
therefore in
% any scale no contrast change > epsilon will not have been found
in the
% loop above.
% we manually need to assign them the biggest possible scale.

idx = find(sm == 0);
sm(idx) = 8;

v1final = zeros(size(v,1),size(v,2));

% build the local luminance map with luminance values taken from
the
% neighborhoods with appropriate scale

for x = 1:size(v1,1)
    for y = 1:size(v1,2)
        v1final(x,y) = v1(x,y,sm(x,y));
    end
end

% display luminance with a/key as in global operator
delta = 0.0001;
a = 0.72;
numpixels = size(hdr,1) * size(hdr,2);
key = exp((1/numpixels)*(sum(sum(log(delta + v1final)))));
scaledluminance = v1final * (a/key);
luminancelocal    =    (luminancemap    *    (a/key))    ./    (1    +
scaledluminance);

% display luminance (default)
% luminancelocal = luminancemap ./ (1 + v1final);

% re-apply color according to Fattals paper "Gradient Domain High
Dynamic
% Range Compression"
ldrlocal = zeros(size(hdr));
for i = 1:3
```

```
    % hdr(:,:,i) ./ luminance) must be between 0 and 1
    % but hdr often contains bigger values than luminance
    % so the resulting ldr image needs to be clamped

    ldrlocal(:,:,i) = ((hdr(:,:,i) ./ luminancemap) .^ saturation)
.* luminancelocal;

end

% clamp resulting LDR image to 1
indices = find(ldrlocal > 1);
ldrglobal(indices) = 1;

% end timer
time = toc;

% display
imshow(ldrlocal);

% write
% imwrite(ldrlocal, 'output bristolb local_reinhard.jpg');
% imwrite(ldrlocal, 'output memorial local_reinhard.jpg');
% imwrite(ldrlocal, 'output rosette local_reinhard.jpg');

imwrite(ldrlocal, 'output book_matlab local_reinhard.jpg');
% imwrite(ldrlocal, 'output lab1_matlab local_reinhard.jpg');
% imwrite(ldrlocal, 'output lab2_matlab local_reinhard.jpg');
% imwrite(ldrlocal, 'output window_matlab local_reinhard.jpg');

% imwrite(ldrlocal, 'output book_pdjm local_reinhard.jpg');
% imwrite(ldrlocal, 'output lab1_pdjm local_reinhard.jpg');
% imwrite(ldrlocal, 'output lab2_pdjm local_reinhard.jpg');
% imwrite(ldrlocal, 'output window_pdjm local_reinhard.jpg');

% imwrite(ldrlocal, 'output book_photomatix local_reinhard.jpg');
% imwrite(ldrlocal, 'output lab1_photomatix local_reinhard.jpg');
% imwrite(ldrlocal, 'output lab2_photomatix local_reinhard.jpg');
%        imwrite(ldrlocal,        'output        window_photomatix
local_reinhard.jpg');
```

APPENDIX R

MATLAB File For Garrett et al Method (main_icam.m)

(main_icam.m)

```
%% Garrett et al Method
% main: illustrates how to use the iCAM HDR tone mapping
% input: HDR image with *.hdr extension
% output: tonemapped HDR image with *.jpg extension
% based on the program written by Lawrence Taplin and Garrett M.
Johnson

% Modified by:
% Wazirah Md Esa

% start timer
tic;

% read and load HDR image
% img = hdrread('input bristolb.hdr');
% img = hdrread('input memorial.hdr');
% img = hdrread('input rosette.hdr');

img = hdrread('input book_matlab.hdr');
% img = hdrread('input lab1_matlab.hdr');
% img = hdrread('input lab2_matlab.hdr');
% img = hdrread('input window_matlab.hdr');

% img = hdrread('input book_pdjm.hdr');
% img = hdrread('input lab1_pdjm.hdr');
% img = hdrread('input lab2_pdjm.hdr');
% img = hdrread('input window_pdjm.hdr');

% img = hdrread('input book_photomatix.hdr');
% img = hdrread('input lab1_photomatix.hdr');
% img = hdrread('input lab2_photomatix.hdr');
% img = hdrread('input window_photomatix.hdr');

hdr = double(img);

% assumption: the image don't actually have physical radiance
meaning, and
% are normalized to be between 0-1

% normalization
img = img./max(img(:));
green = squeeze(img(:,:,2));
prc = prctile(green(:),99);
scale = 50/prc;

% rendering
hdr_image = render_hdr(img*scale);

% end timer
time = toc;
```

```
% displaying
imshow(hdr_image);

% writing
% imwrite(hdr_image, 'output bristolb exptonemap.jpg');
% imwrite(hdr_image, 'output memorial exptonemap.jpg');
% imwrite(hdr_image, 'output rosette exptonemap.jpg');

imwrite(hdr_image, 'output book_matlab exptonemap.jpg');
% imwrite(hdr_image, 'output lab1_matlab exptonemap.jpg');
% imwrite(hdr_image, 'output lab2_matlab exptonemap.jpg');
% imwrite(hdr_image, 'output window_matlab exptonemap.jpg');

% imwrite(hdr_image, 'output book_pdjm exptonemap.jpg');
% imwrite(hdr_image, 'output lab1_pdjm exptonemap.jpg');
% imwrite(hdr_image, 'output lab2_pdjm exptonemap.jpg');
% imwrite(hdr_image, 'output window_pdjm exptonemap.jpg');

% imwrite(hdr_image, 'output book_photomatix exptonemap.jpg');
% imwrite(hdr_image, 'output lab1_photomatix exptonemap.jpg');
% imwrite(hdr_image, 'output lab2_photomatix exptonemap.jpg');
% imwrite(hdr_image, 'output window_photomatix exptonemap.jpg');
```

APPENDIX S

MATLAB File For Garrett et al Method (render_hdr.m)

(render_hdr.m)

```
%% Garrett et al Method
% render_hdr: performs iCAM HDR tone mapping
% based on the program written by Garrett M. Johnson

% assumption: RGB image is in a linearized form

function imgout = render_hdr(imgin)

% choose a color space
rgb2xyz = cmatrix('rgb2xyz',2);
% renormalize matrix
rgb2xyz = rgb2xyz./sum(rgb2xyz(:,2)).*100;
% rgb to xyz transform
xyzimg = changecolorspace(imgin, rgb2xyz);

% forward iCAM HDR transform
icamimg = icam_hdr(xyzimg);

% invert IPT transform
icamxyz = inv_ipt(icamimg);

% invert chromatic adaptation transform
icamxyza = inv_cat(icamxyz);

xyz2rgb = cmatrix('xyz2rgb',2);
% renormalize matrix
xyz2rgb = xyz2rgb./sum(xyz2rgb(:,2)).*100;
% xyz to rgb transform
icamrgb = changecolorspace(icamxyza, xyz2rgb);

% find the 0.99 percentile
perc = prctile(icamrgb(:),97); % EXPERIMENT, original value=99

% clip image to the 0.99 percentile
% apply a gamma correction
% scale the min and the max of image to 0 and 255, respectively
imgout = min(icamrgb,perc);
imgout = ((imgout - min(imgout(:))) ./ (max(imgout(:)) -
min(imgout(:)))).^(1/1.7);
imgout = uint8(round(255*imgout));
```

APPENDIX T

MATLAB File For Garrett et al Method (icam_hdr.m)

(icam_hdr.m)

```
%% Garrett et al Method
% icam_hdr: perform iCAM HDR tone mapping
% input: image (xyz)
% output: image (IPT)
% based on the program written by Lawrence Taplin and Garrett
Johnson

function imgipt = icam_hdr(imgin)

% get the "whitepoint" adaptation, which is a blurred version of
the image

sizeimg = size(imgin);
xdim = sizeimg(2);
ydim = sizeimg(1);

distmap = idl_dist(ydim,xdim);

kernel = exp(-1*(distmap./(xdim/4)).^2);
kernel = kernel/kernel(1,1); %added after comparing with ipt_hdr.m

% since we are convolving, normalize the kernel to sum to 1,
% and shift it to center

filter = max(real(fft2(kernel)),0);
filter = filter./filter(1,1);

whitexyz = zeros(size(imgin));

whitexyz(:,:,1) = max(real(ifft2(fft2(imgin(:,:,2)).*filter)),0);
whitexyz(:,:,2) = whitexyz(:,:,1);
whitexyz(:,:,3) = whitexyz(:,:,1);

% perform the HDR chromatic adaptation transform
imgcat = cat_hdr(imgin, whitexyz);

% transform into the IPT color space, blurred appropriately
imgipt = ipt_hdr(imgcat, squeeze(imgin(:,:,2)));
```

APPENDIX U

MATLAB File For Garrett et al Method (cat_hdr.m)

(cat_hdr.m)

```
%% Garrett et al Method
% cat_hdr: perform a HDR chromatic adaptation transform for the
% iCAM image appearance model
% notes: this is not much a chromatic adaptation as a luminance
adaptation,
% since we are normalizing all the tristimulus values with a
single
% number Y
% based on the program written by Lawrence Taplin and Garrett
Johnson

function xyzadapt = cat_hdr(image, whitepoint)

% define the xyz to rgb transform matrix, using CIECAM02 transform
MCAT02 = [0.7328 0.4296 -0.1624; -0.7036 1.6975 0.0061; 0.0030
0.0136 0.9834];

MCAT02i = inv(MCAT02);

% set the whitepoint for D65, as that is where IPT is defined
% average sunlight
xyz_d65 = [95.05 100.00 108.88];

rgb_d65 = changecolorspace(xyz_d65, MCAT02);
rgb_img = changecolorspace(image, MCAT02);
rgb_white = changecolorspace(whitepoint, MCAT02);

% normalize rgb_white
rgb_white = rgb_white * max(rgb_img(:)) / max(rgb_white(:));

% manually set a degree of adaptation
% 0 for no adaptation to adopted whitepoint,
% 1 for complete adaptation to adopted whitepoint,
% not less than 0.65 for a dark surround,
% exponentially converge to 1 for average surround
D = 1.0;

rc = (D * rgb_d65(1) ./ rgb_white(:,:,1) + 1 - D) .*
rgb_img(:,:,1);
gc = (D * rgb_d65(2) ./ rgb_white(:,:,2) + 1 - D) .*
rgb_img(:,:,2);
bc = (D * rgb_d65(3) ./ rgb_white(:,:,3) + 1 - D) .*
rgb_img(:,:,3);

sizeimg = size(image);
xdim = sizeimg(2);
ydim = sizeimg(1);
```

```
imageadapt = zeros(ydim, xdim, 3);

imageadapt(:,:,1) = rc;
imageadapt(:,:,2) = gc;
imageadapt(:,:,3) = bc;

xyzadapt = changecolorspace(imageadapt, MCAT02i);
```

APPENDIX V

MATLAB File For Garrett et al Method (ipt_hdr.m)

(ipt_hdr.m)

```
%% Garrett et al Method
% ipt_hdr: performs the ipt transform for the iCAM model, it
% creates a low-pass image mask based on the Y channel, and uses
the
% CIECAM02 surround formula to modify the IPT exponent
% based on the program written by Garrett Johnson

function imageipt = ipt_hdr(xyzimage, imagey)

sizeimg = size(xyzimage);
xdim = sizeimg(2);
ydim = sizeimg(1);

distmap = idl_dist(ydim, xdim);

% the kernel is a Gaussian function of width xdim/3.0
kernel = exp(-1*(distmap./(xdim/1)).^2);
kernel = kernel/kernel(1,1);

% transfrom Gaussian to frequency domain
% then, normalize the DC component
filter = max(real(fft2(kernel)),0);
filter = filter/filter(1,1);

% filter the image
ylow = max(real(ifft2(fft2(imagey).*filter)),0);

xyz2lms = cmatrix('xyz2lms');
lmsimage = changecolorspace(xyzimage/100.0, xyz2lms);

% the exponent scale is calculated based on the surround function
from
% CIECAM02
% It is set to 1.0 for a value of 100.0
exp_scale = (1/1.7) * (0.2*(1./(5*ylow + 1)).^4 .* (5*ylow) +
0.1*(1-(1./(5*ylow+1)).^4).^2 .* (5*ylow).^(1/3));

lmsimagex = lmsimage;

% apply the IPT exponent along with the scaling factor
lmsimagex(:,:,1) = abs(lmsimage(:,:,1)).^(exp_scale*0.43);
lmsimagex(:,:,2) = abs(lmsimage(:,:,2)).^(exp_scale*0.43);
lmsimagex(:,:,3) = abs(lmsimage(:,:,3)).^(exp_scale*0.43);

iptmat = [0.4000 0.4000, 0.2000; 4.4550 -4.8510 0.3960; 0.8056
0.3572 -1.1628];
imageipt = changecolorspace(lmsimagex, iptmat);
```

APPENDIX W

MATLAB File For Garrett et al Method (inv_ipt.m)

(inv_ipt.m)

```
%% Garrett et al Method
% inv_ipt: inverts the ipt transform for display, uses a single
%  number  (0.43)  for  the  inversion  rather  than  a  spatially
localized
% low-pass mask
% based on the program written by Lawrence Taplin and Garrett M.
Johnson

function xyzimage = inv_ipt(iptimage)

inviptmat  =  inv([0.4000  0.4000  0.2000;  4.4550  -4.8510  0.3960;
0.8056 0.3572 -1.1628]);

lms2xyz = cmatrix('lms2xyz');

lmsimage = changecolorspace(iptimage, inviptmat);

xyzimage = changecolorspace(abs(lmsimage).^(1/.43), lms2xyz);
```

APPENDIX X

MATLAB File For Garrett et al Method (inv_cat.m)

(inv_cat.m)

```
%% Garrett et al Method
% inv_cat: inverts the chromatic adaptation transform from D65 to
% the "monitor" white so it can be displayed on the monitor, note
that in
% the iCAM framework the inverse transform is done with a single
number
% based on the program written by: Lawrence Taplin and Garrett M.
Johnson

function xyzadapt  = inv_cat(imgin)

% define the xyz to rgb transform matrix, using CIECAM97 transform
% taken from original implementation
% MCAT02 = [0.8562 0.3372 -0.1934; -0.8360 1.8327 0.0033; 0.0357 -
0.0469
% 1.0112];

% define the xyz to rgb transform matrix, using CIECAM02 transfrom
MCAT02 = [0.7328 0.4296 -0.1624; -0.7036 1.6974 0.0061; 0.0030
0.0136 0.9834];

MCAT02i = inv(MCAT02);

% average sunlight
xyz_d65 = [95.05 100.00 108.88];

whitepoint = [100.0 100.0 100.0];

rgb_img = changecolorspace(imgin, MCAT02);
rgb_d65 = changecolorspace(xyz_d65, MCAT02 );
rgb_white = changecolorspace(whitepoint, MCAT02);

% manually set a degree of adaptation
% 0 for no adaptation to adopted whitepoint,
% 1 for complete adaptation to adopted whitepoint,
% not less than 0.65 for a dark surround,
% exponentially converge to 1 for average surround
D = 0.10;

rc = (D * rgb_white(1) ./ rgb_d65(1) + 1 - D) .* rgb_img(:,:,1);
gc = (D * rgb_white(2) ./ rgb_d65(2) + 1 - D) .* rgb_img(:,:,2);
bc = (D * rgb_white(3) ./ rgb_d65(3) + 1 - D) .* rgb_img(:,:,3);

sizeimg = size(imgin);
xdim = sizeimg(2);
ydim = sizeimg(1);

imageadapt = zeros(ydim, xdim, 3);
```

```
imageadapt(:,:,1) = rc;
imageadapt(:,:,2) = gc;
imageadapt(:,:,3) = bc;

xyzadapt = changecolorspace(imageadapt, MCAT02i);
```

APPENDIX Y

MATLAB File For Garrett et al Method (changecolorspace.m)

(changecolorspace.m)

```
%% Garrett et al Method
% changecolorspace: converts column vectors in the output image
% representation into column vectors in the output representation,
% ensures the input image is put back into the same format as it
was passed

% based on scielab procedure of Wandell and Zhang
% based on the program written by Lawrence Taplin and Garrett
Johnson

function imgout = changecolorspace(imgin, colormatrix)

sizeimgin = size(imgin);

% put the pixels in the input image into the rows of a very large
matrix
if length(sizeimgin) == 3
    imgin     =     reshape(imgin,     sizeimgin(1)*sizeimgin(2),
sizeimgin(3));
end

% multiply by color matrix to convert the pixels to the output
color space
imgout = imgin*colormatrix;

% put the output image to the shape used before
if length(sizeimgin) == 3
    imgin    =    reshape(imgin,    sizeimgin(1),    sizeimgin(2),
sizeimgin(3));
    imgout    =    reshape(imgout,    sizeimgin(1),    sizeimgin(2),
sizeimgin(3));
end
```

APPENDIX Z

MATLAB File For Garrett et al Method (cmatrix.m)

(cmatrix.m)

```
%% Garrett et al Method
% cmatrix: returns a 3x3 color matrix used by changecolorspace
% program
% based on the scielab code from Wandell and Zhang
% based on the program written by Lawrence Taplin and Garrett
Johnson

% matrixtype has the following options:
%    'lms2xyz' -- Hunt-Pointer-Estevez transformation from cone
%                 to XYZ, normalized for D65 (lms=[100 100 100]
for D65).
%    'xyz2lms' -- inverse of lms2xyz.
%    'rgb2xyz' -- rgb to xyz 2 degree.
%    'xyz2rgb' -- inverse of the above matrix
%
% spacetype specifies what type of xyz space (CIE1931 2 degree or
% CIE1964 10 degree) is required.
%    spacetype = 2: cie1931 2 degree XYZ   (default)
%    spacetype = 10: cie1964 10 degree XYZ


function result = cmatrix(matrixtype, spacetype)

if (nargin ~= 2)
    spacetype = 2;
end

switch matrixtype

    case 'lms2xyz', % we use this
        result  =  inv([0.4002  0.7077  -0.0807;  -0.2280  1.1500
0.0612; 0.0 0.0 0.9184]);

    case 'xyz2lms', % we use this
        result = [0.4002 0.7077 -0.0807; -0.2280 1.1500 0.0612;
0.0 0.0 0.9184];

    case 'rgb2xyz', % we use this
        if (spacetype == 2)
            result = [41.384 22.155 0.487; 25.053 51.424 5.438;
11.014 9.743 56.089];
        else
            result = [17.4665 27.7468 16.5398; 10.0969 48.1835
11.6466; 0.9293 7.3710 85.5683];
        end

    case 'xyz2rgb', % we use this
        if (spacetype == 2)
            result  =  inv([41.384  22.155  0.487;  25.053  51.424
5.438; 11.014 9.743 56.089]);
```

```
        else
            result = inv([17.4665 27.7468 16.5398; 10.0969 48.1835
11.6466; 0.9293 7.3710 85.5683]);
        end

    otherwise
        result = 0;
end

% result = result';
```

APPENDIX AA

MATLAB File For Garrett et al Method (idl_dist.m)

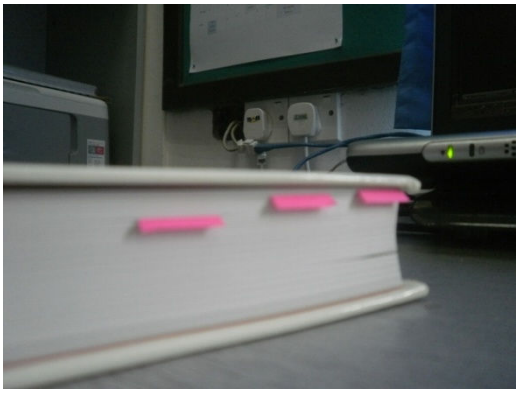
(idl_dist.m)

```
%% Garrett et al Method
% idl_dist: a direct port of the IDL distance function
% based on the program written by Lawrence Taplin

function a = idl_dist(m,n)

x = 0:(n-1);
x = min(x, (n-x)).^2;

if nargin == 1
    m = n;
end

a = zeros(m,n);

for i = 0:m/2
    y = sqrt(x + i.^2);
    a(i+1,:) = y;
    if i ~= 0
        a(m-i+1,:) = y;
    end
end
```

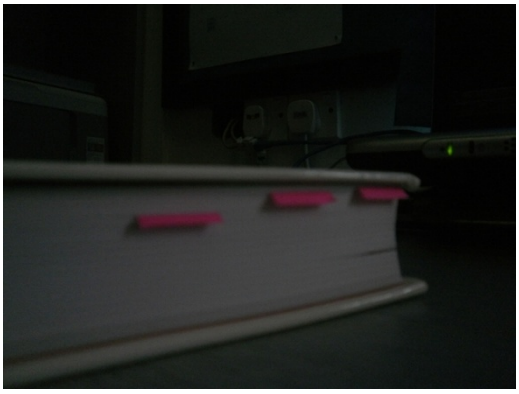
APPENDIX BB

LDR Source Images (book)

Exposure time = 1/20 second



Exposure time = 1/125 second



Exposure time = 1/640 second

APPENDIX CC

LDR Sources Images (lab1)

Exposure time = 1/50 second

Exposure time = 1/100

Exposure time = 1/250

APPENDIX DD

LDR Sources Images (lab2)

Exposure time = 1/50 second

Exposure time = 1/1000 second

Exposure time = 1/250 second

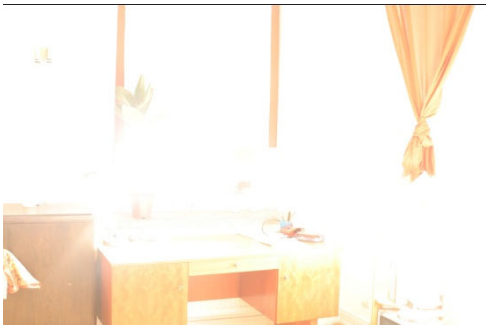
Exposure time = 1/2500 second

Exposure time = 1/700

Exposure time = 1/10000 second

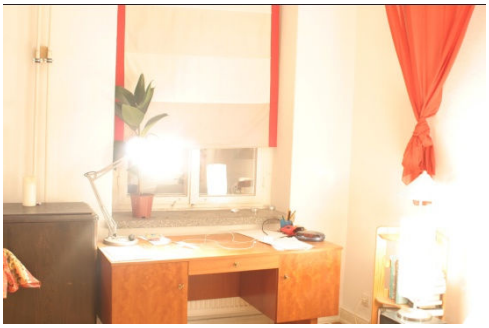
APPENDIX EE

LDR Source Images (window)



Exposure time = 15 seconds

Exposure time = 1/4 second



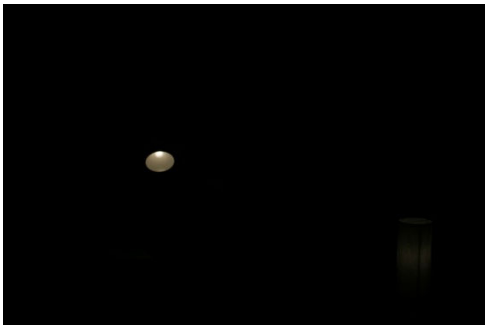
Exposure time = 4 seconds

Exposure time = 1/15 second

Exposure time = 1 second

Exposure time = 1/60 second

Exposure time = 1/250 second

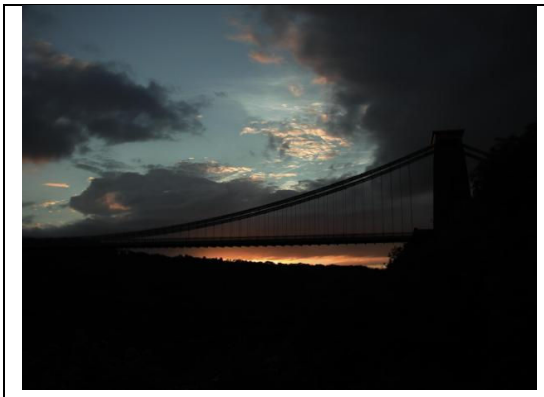


Exposure time = 1/1000 second

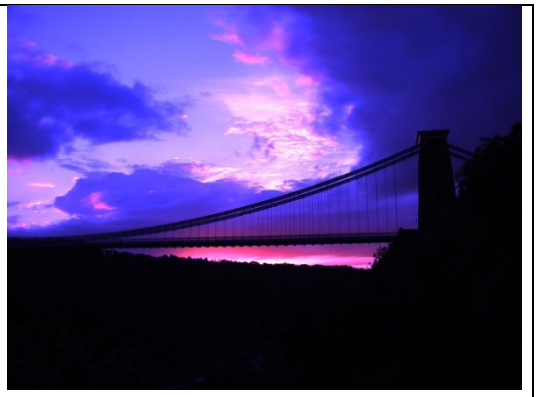
Exposure time = 1/4000 second

APPENDIX FF

Output Images of Set 1 (bristolb)

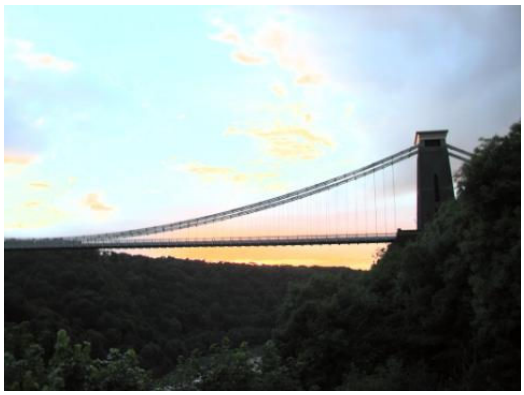


Logarithmic

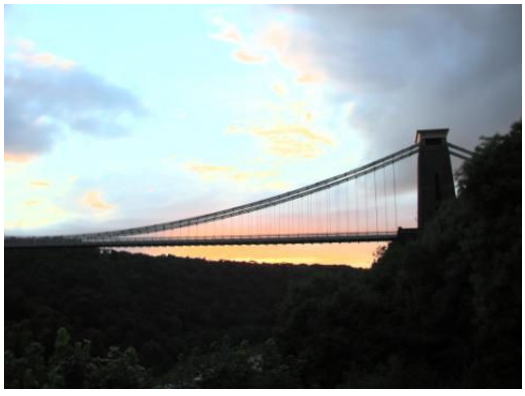


Modified Logarithmic

Exponential (δ=0.0001)



Exponential (δ=0.001)



Exponential (δ=0.01)

Exponential (δ=0.1)

Modified Exponential (δ=0.0001)

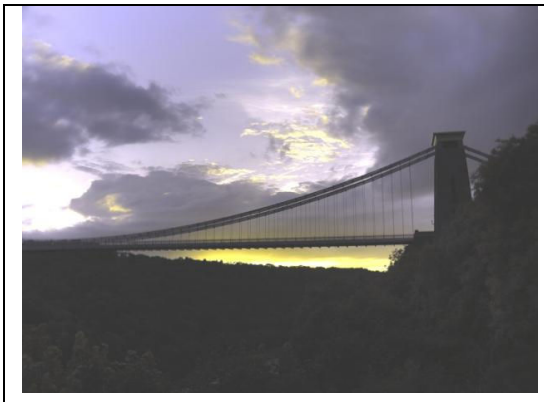
Modified Exponential (δ=0.001)

Modified Exponential (δ=0.01)

Modified Exponential (δ=0.1)

Reinhard et al Global Operator

Reinhard et al Local Operator



Garrett et al

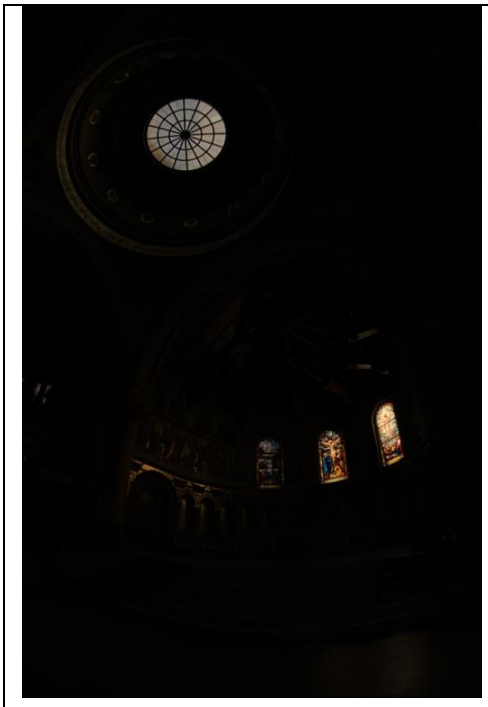
MATLAB “tonemap”

Photomatix Details Enhancer

Photomatix Tone Compressor

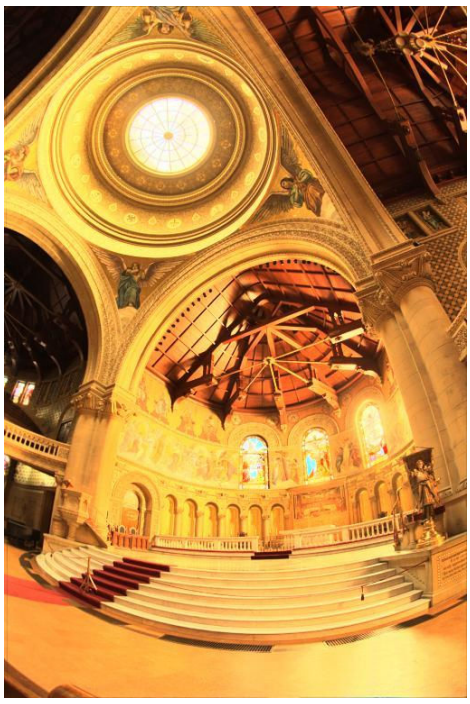
APPENDIX GG

Output Images of Set 1 (memorial)

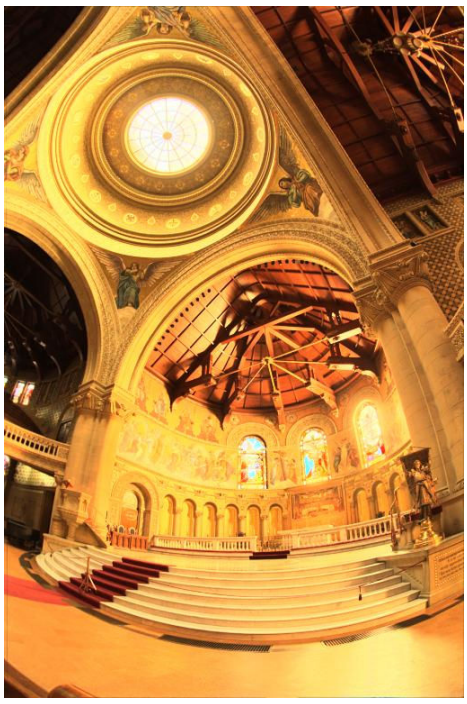


Logarithmic

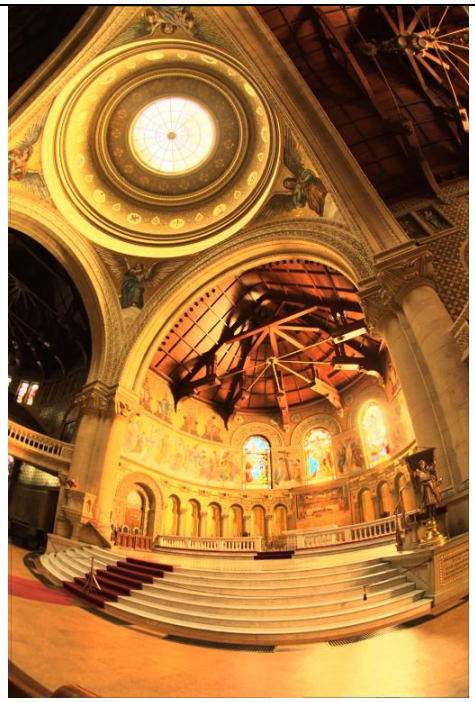
Modified Logarithmic



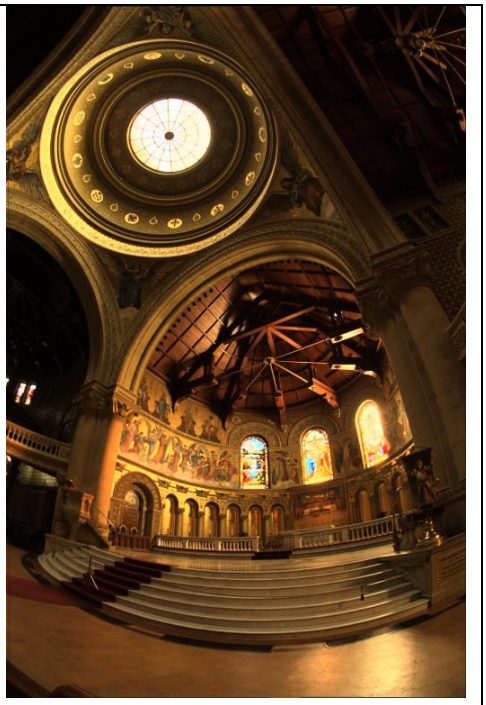
Exponential (δ=0.0001)



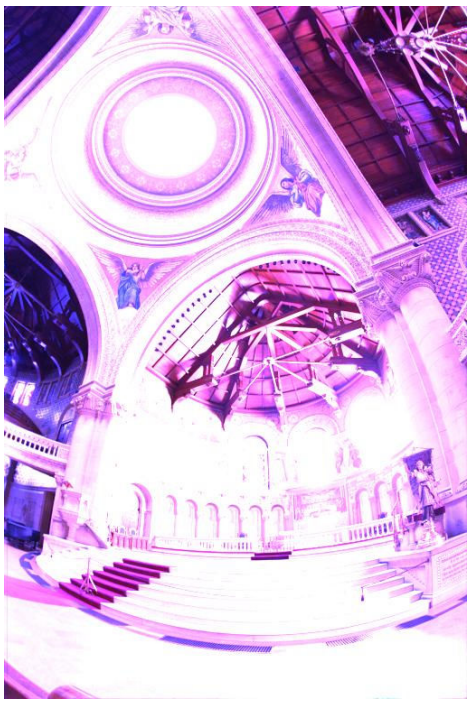
Exponential (δ=0.001)



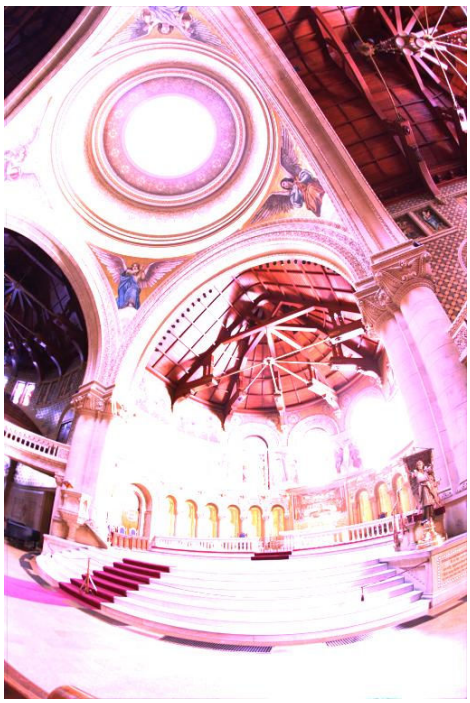
Exponential (δ=0.01)



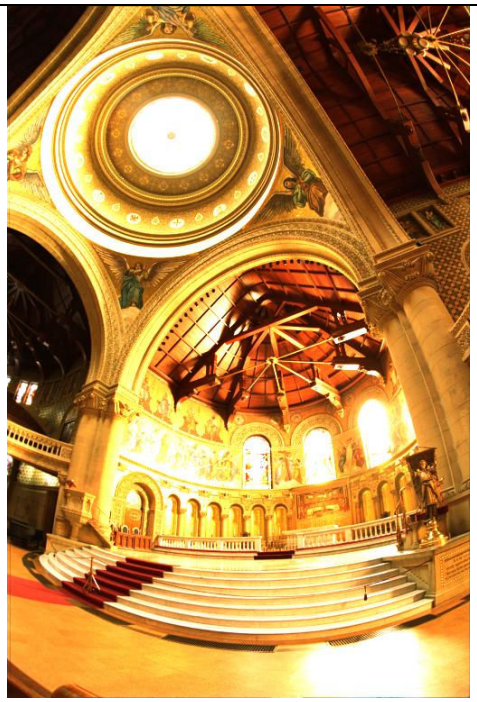
Exponential (δ=0.1)



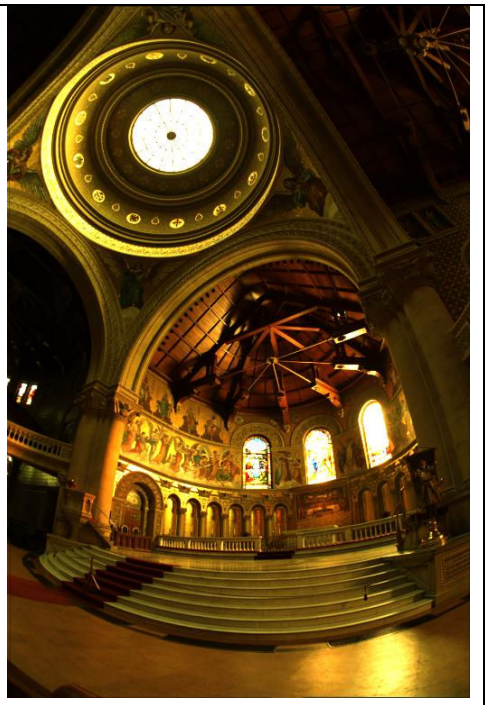
Modified Exponential (δ=0.0001)



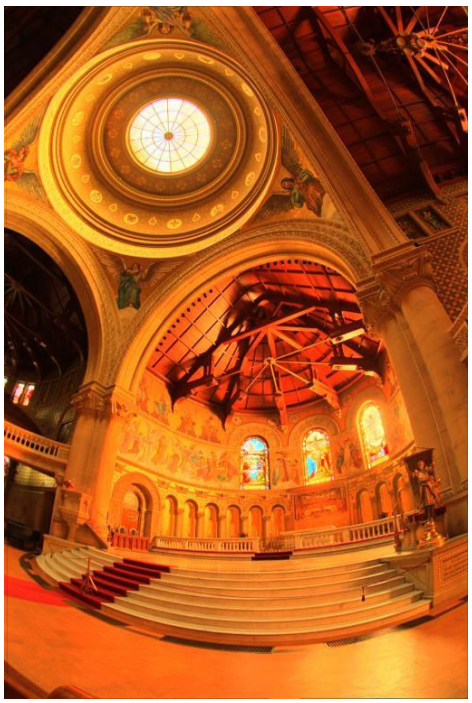
Modified Exponential (δ=0.001)



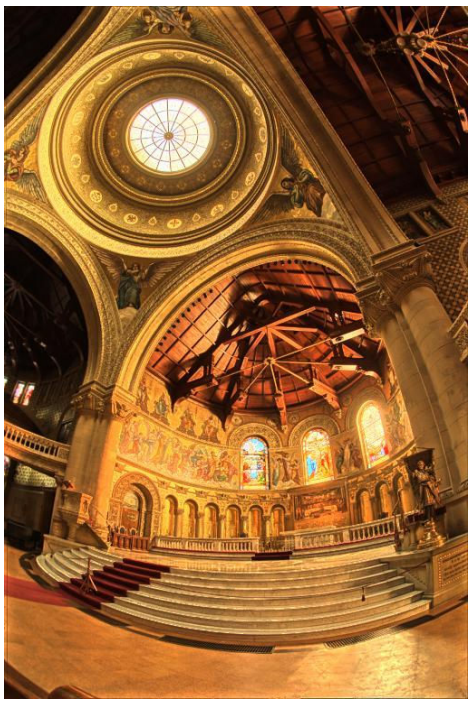
Modified Exponential (δ=0.01)



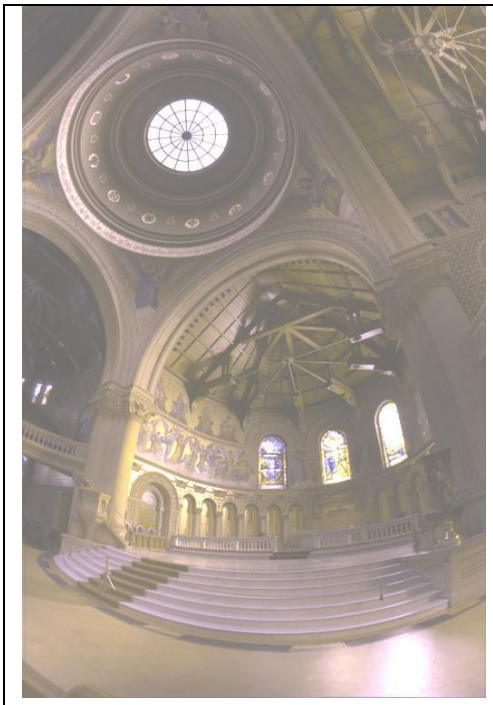
Modified Exponential (δ=0.1)



Reinhard et al Global Operator

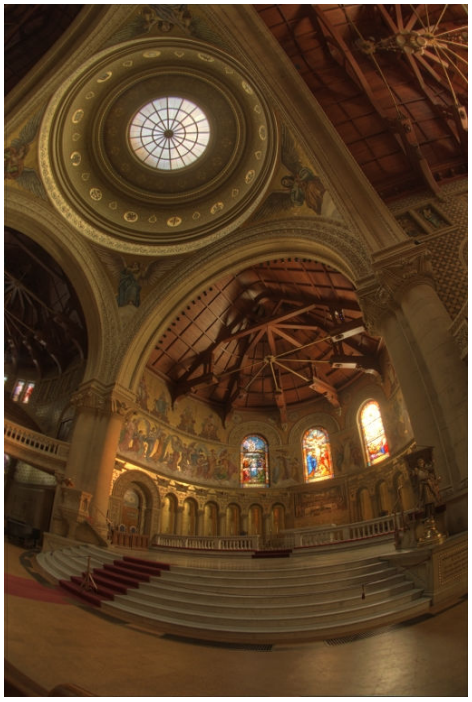


Reinhard et al Local Operator

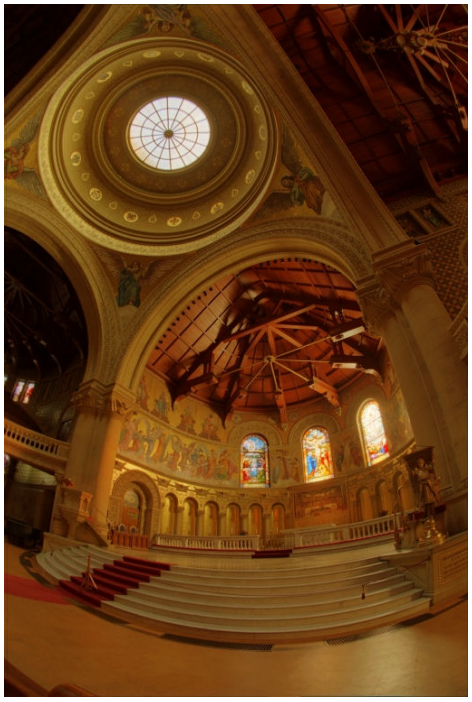


Garrett et al

MATLAB “tonemap”



Photomatix Details Enhancer

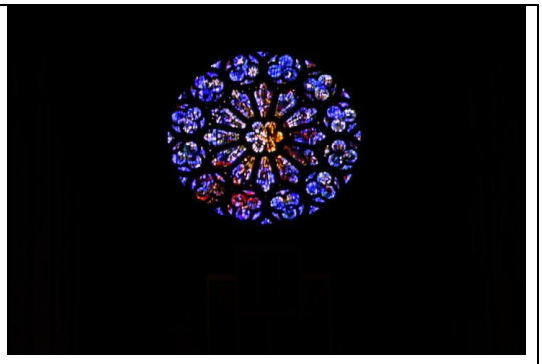


Photomatix Tone Compressor

APPENDIX HH

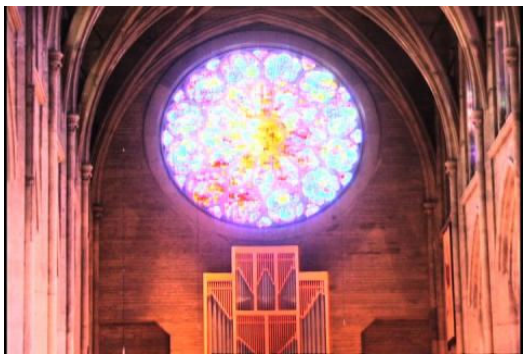
Output Images of Set 1 (rosette)

Logarithmic

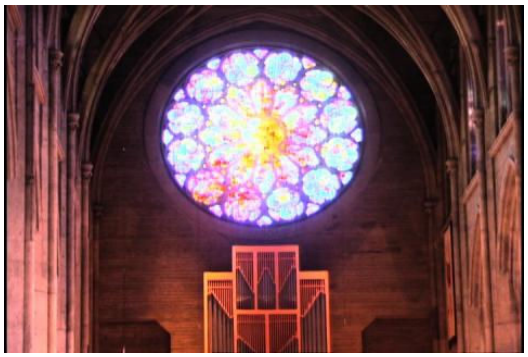


Modified Logarithmic

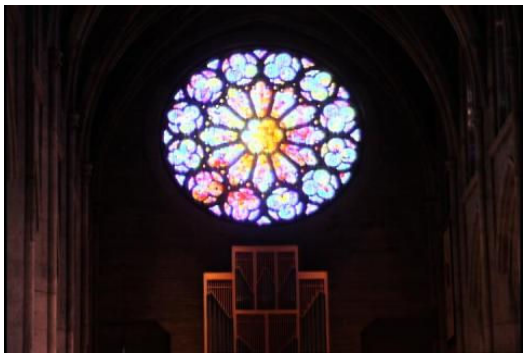
Exponential (δ=0.0001)



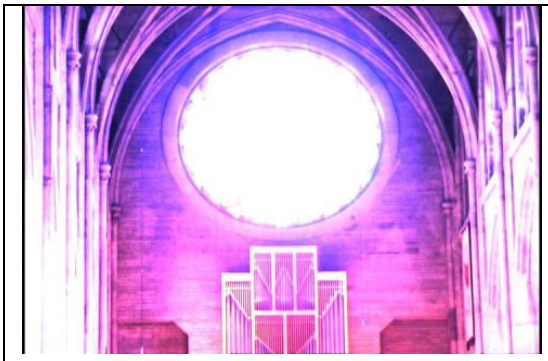
Exponential (δ=0.001)



Exponential (δ=0.01)



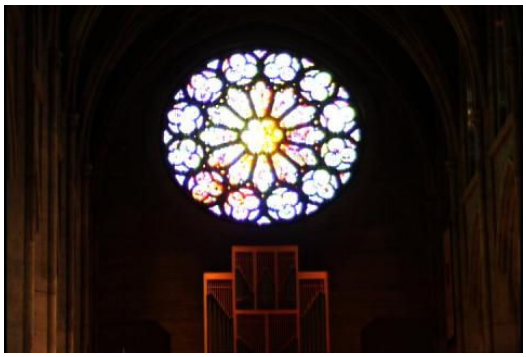
Exponential (δ=0.1)



Modified Exponential (δ=0.0001)

Modified Exponential (δ=0.001)

Modified Exponential (δ=0.01)



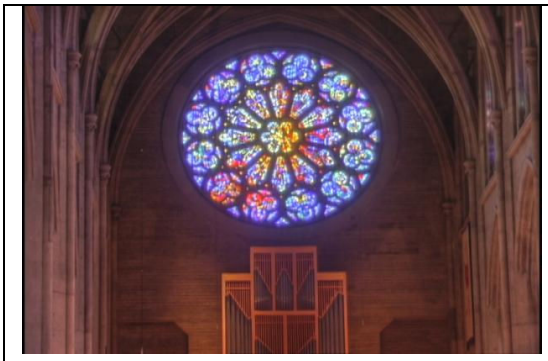
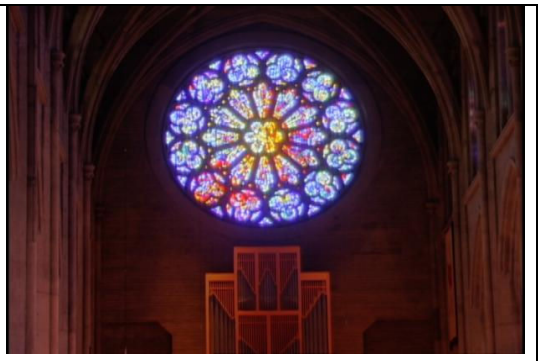
Modified Exponential (δ=0.1)

Reinhard et al Global Operator

Reinhard et al Local Operator

Garrett et al

MATLAB “tonemap”

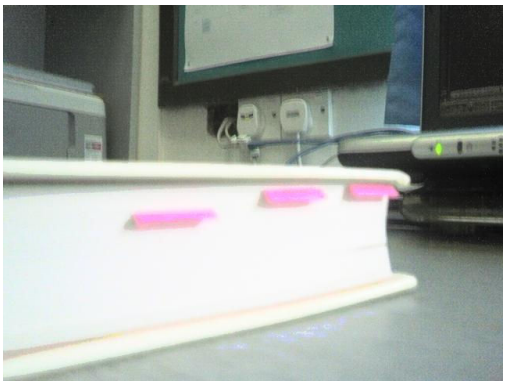
	
Photomatix Details Enhancer	Photomatix Tone Compressor

APPENDIX II

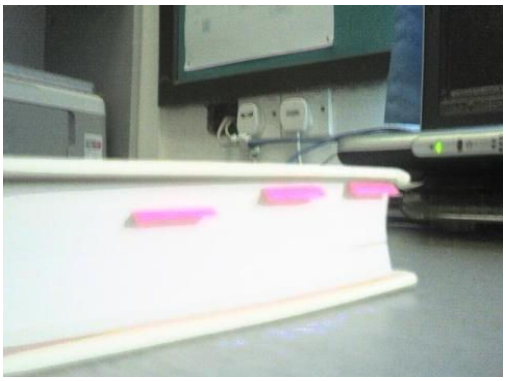
Output Images of Set 2 (book)

Logarithmic

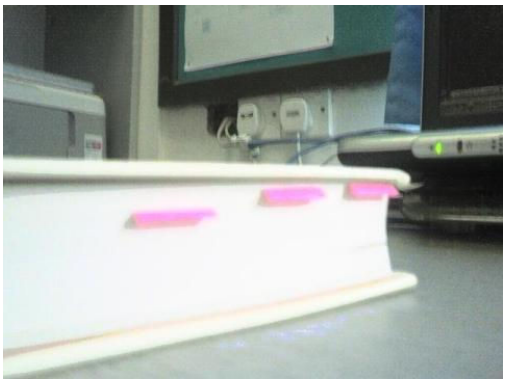
Modified Logarithmic



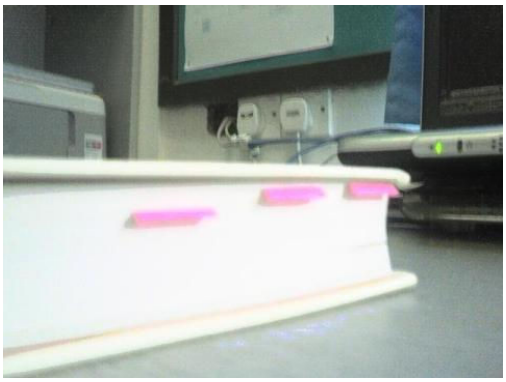
Exponential (δ=0.0001)



Exponential (δ=0.001)



Exponential (δ=0.01)



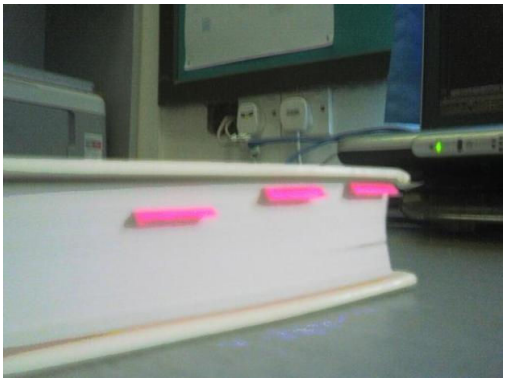
Exponential (δ=0.1)

Modified Exponential (δ=0.0001)

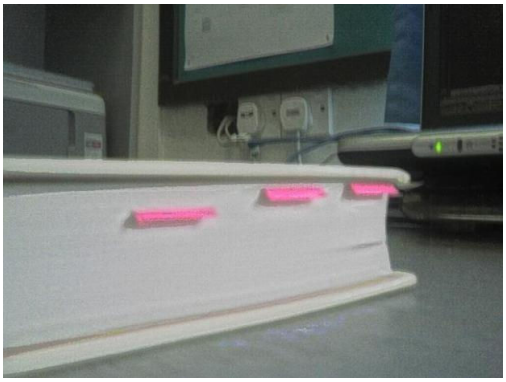
Modified Exponential (δ=0.001)

Modified Exponential (δ=0.01)

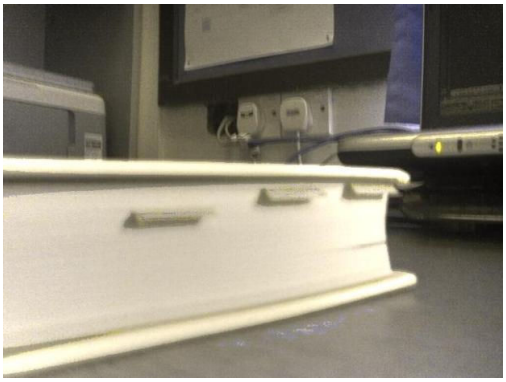
Modified Exponential (δ=0.1)



Reinhard et al Global Operator

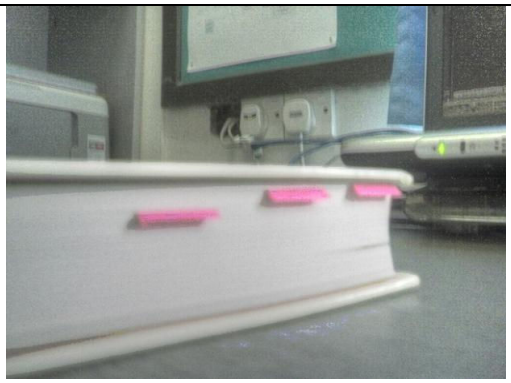
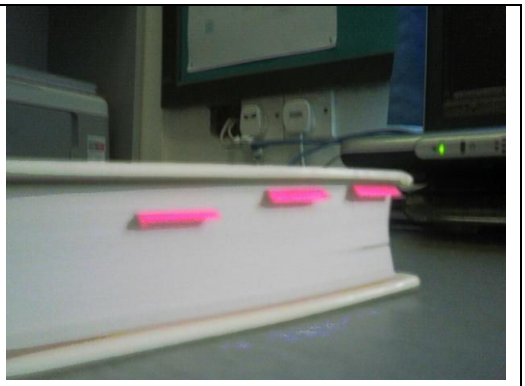


Reinhard et al Local Operator



Garrett et al

MATLAB “tonemap”

	
Photomatix Details Enhancer	Photomatix Tone Compressor

APPENDIX JJ

Output Images of Set 2 (lab1)

Logarithmic

Modified Logarithmic

Exponential (δ=0.0001)

Exponential (δ=0.001)

Exponential (δ=0.01)

Exponential (δ=0.1)

Modified Exponential (δ=0.0001)

Modified Exponential (δ=0.001)

Modified Exponential (δ=0.01)

Modified Exponential (δ=0.1)

Reinhard et al Global Operator

Reinhard et al Local Operator

Garrett et al

MATLAB “tonemap”

Photomatix Details Enhancer

Photomatix Tone Compressor

APPENDIX KK

Output Images of Set 2 (lab2)

Logarithmic

Modified Logarithmic

Exponential (δ=0.0001)

Exponential (δ=0.001)

Exponential (δ=0.01)

Exponential (δ=0.1)

Modified Exponential (δ=0.0001)

Modified Exponential (δ=0.001)

Modified Exponential (δ=0.01)

Modified Exponential (δ=0.1)

Reinhard et al Global Operator

Reinhard et al Local Operator

Garrett et al

MATLAB “tonemap”

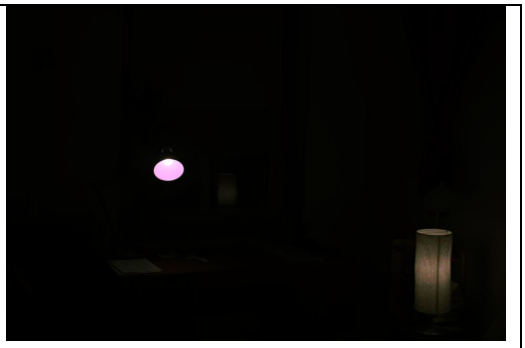
Photomatix Details Enhancer

Photomatix Tone Compressor

APPENDIX LL

Output Images of Set 2 (window)

Logarithmic



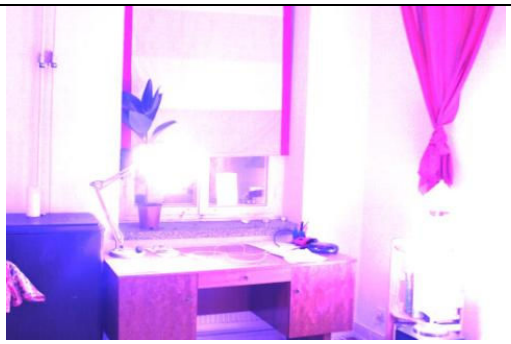
Modified Logarithmic

Exponential (δ=0.0001)

Exponential (δ=0.001)

Exponential (δ=0.01)

Exponential (δ=0.1)



Modified Exponential (δ=0.0001)

Modified Exponential (δ=0.001)

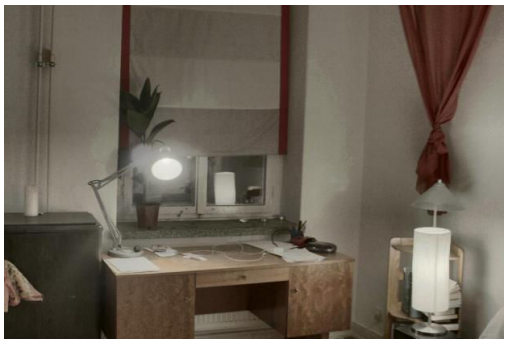
Modified Exponential (δ=0.01)

Modified Exponential (δ=0.1)

Reinhard et al Global Operator

Reinhard et al Local Operator

Garrett et al

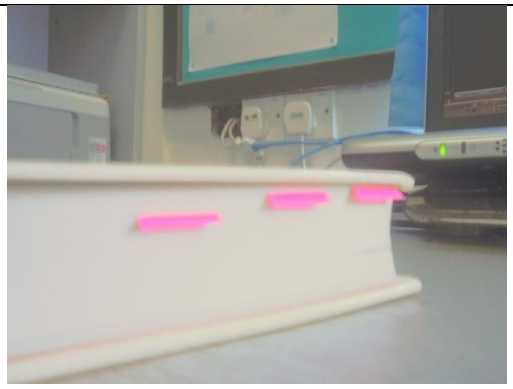


MATLAB “tonemap”

	
Photomatix Details Enhancer	Photomatix Tone Compressor

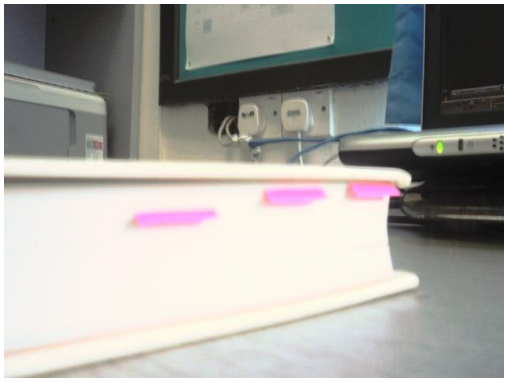
APPENDIX MM

Output Images of Set 3 (book)

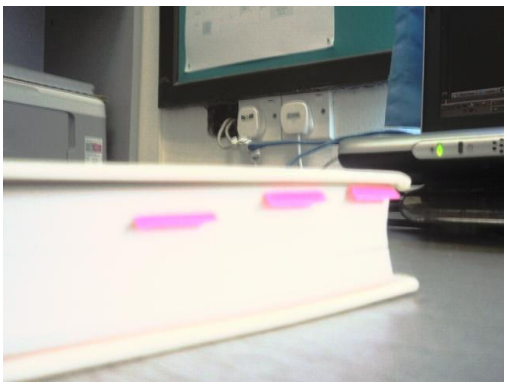


Logarithmic

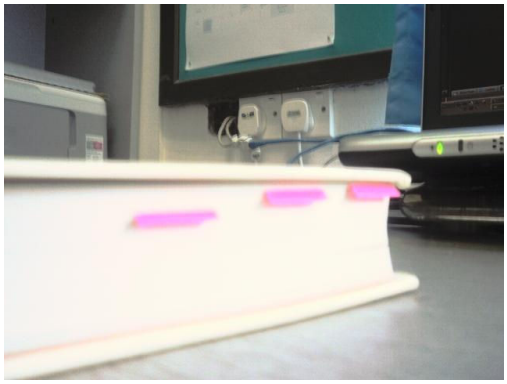
Modified Logarithmic



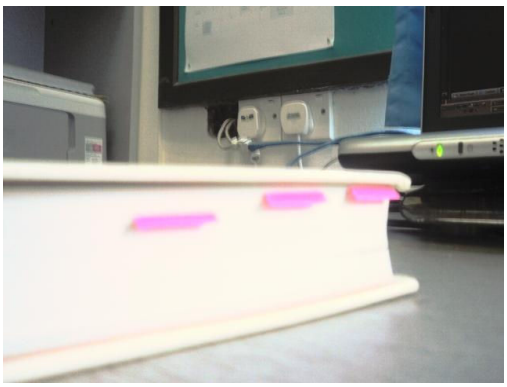
Exponential (δ=0.0001)



Exponential (δ=0.001)

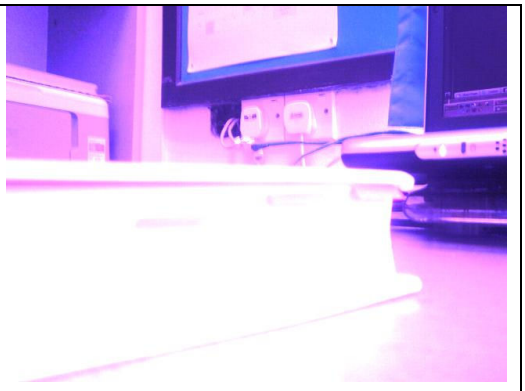


Exponential (δ=0.01

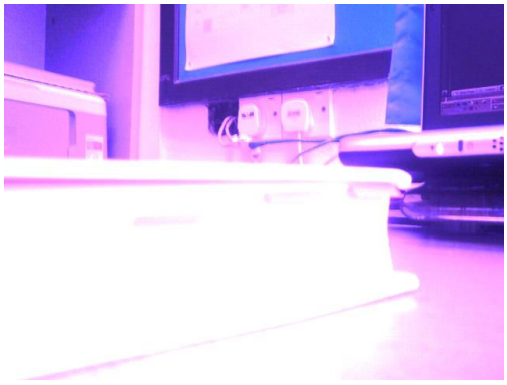


Exponential (δ=0.1)

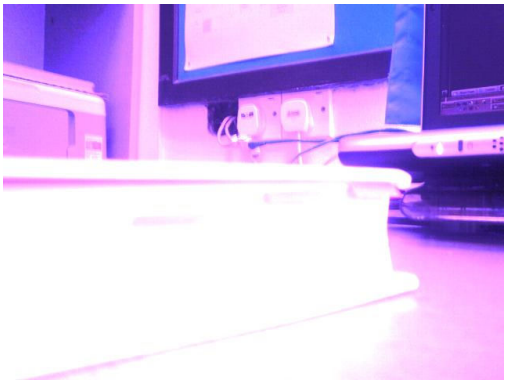
Modified Exponential (δ=0.0001)



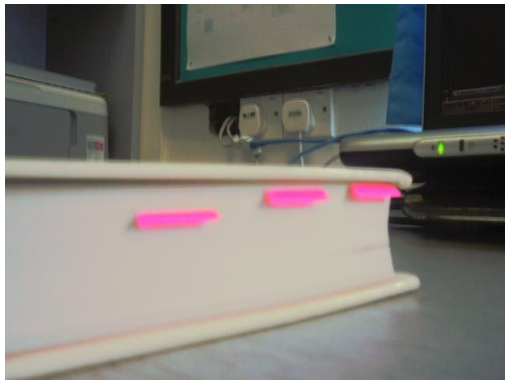
Modified Exponential (δ=0.001)



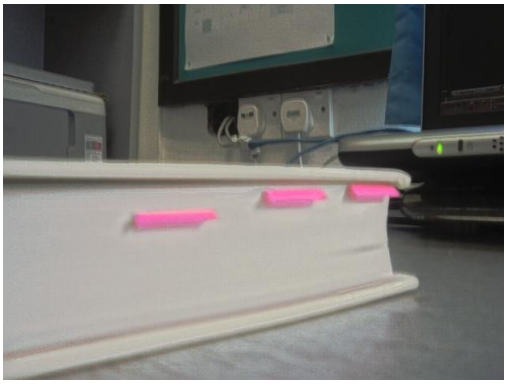
Modified Exponential (δ=0.01)



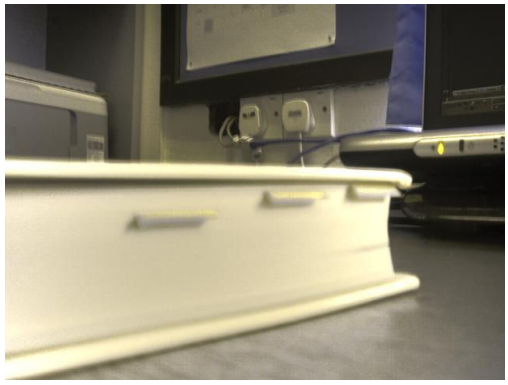
Modified Exponential (δ=0.1)



Reinhard et al Global Operator

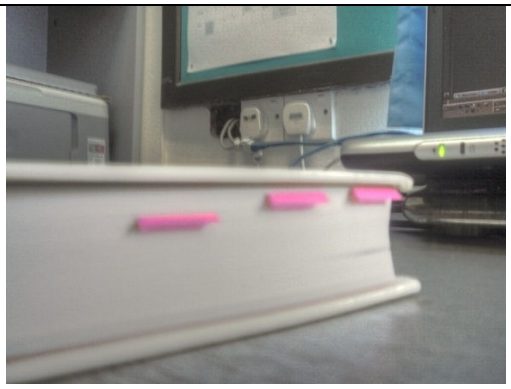


Reinhard et al Local Operator



Garrett et al

MATLAB “tonemap”

	
Photomatix Details Enhancer	Photomatix Tone Compressor

APPENDIX NN

Output Images of Set 3 (lab1)

Logarithmic

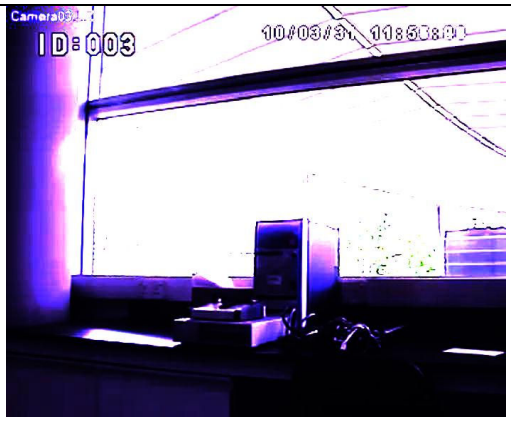
Modified Logarithmic

Exponential (δ=0.0001)

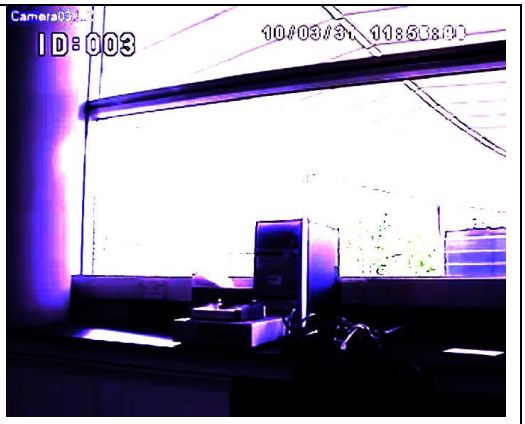
Exponential (δ=0.001)

Exponential (δ=0.01)

Exponential (δ=0.1)



Modified Exponential (δ=0.0001)



Modified Exponential (δ=0.001)

Modified Exponential (δ=0.01)

Modified Exponential (δ=0.1)

Reinhard et al Global Operator

Reinhard et al Local Operator

Garrett et al

MATLAB “tonemap”

Photomatix Details Enhancer

Photomatix Tone Compressor

APPENDIX OO

Output Images of Set 3 (lab2)

Logarithmic

Modified Logarithmic

Exponential (δ=0.0001)

Exponential (δ=0.001)

Exponential (δ=0.01)

Exponential (δ=0.1)

Modified Exponential (δ=0.0001)

Modified Exponential (δ=0.001)

Modified Exponential (δ=0.01)

Modified Exponential (δ=0.1)

Reinhard et al Global Operator

Reinhard et al Local Operator

Garrett et al

MATLAB “tonemap”

Photomatix Details Enhancer

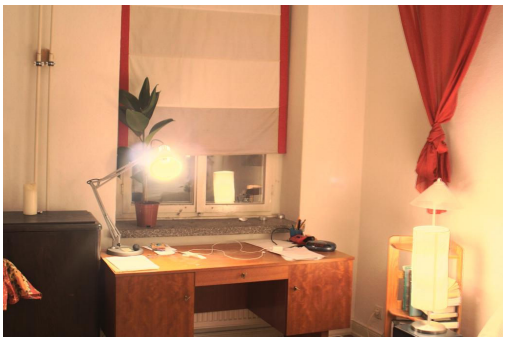
Photomatix Tone Compressor

APPENDIX PP

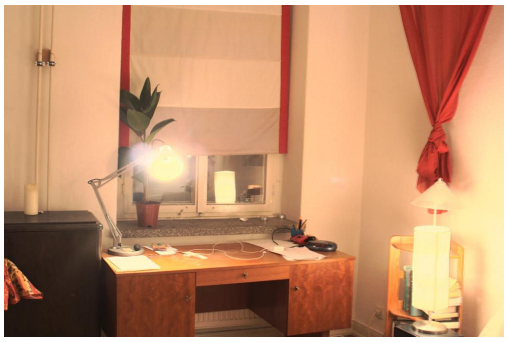
Output Images of Set 3 (window)

Logarithmic

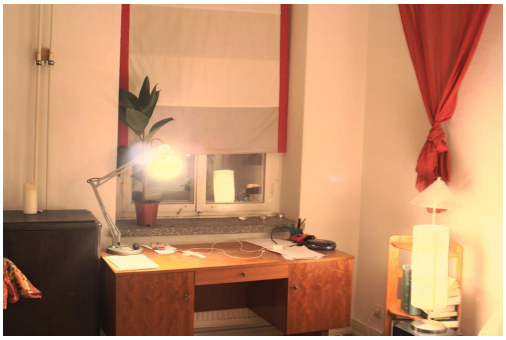
Modified Logarithmic



Exponential (δ=0.0001)

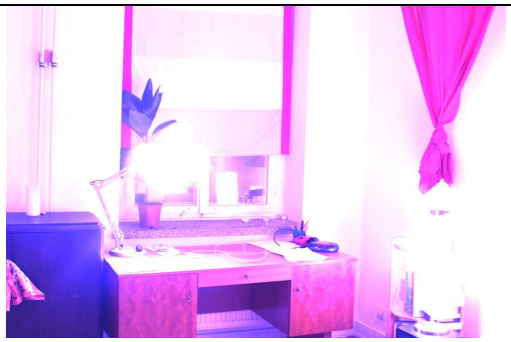


Exponential (δ=0.001)

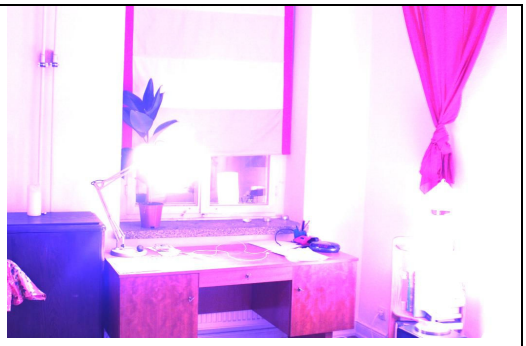


Exponential (δ=0.01)

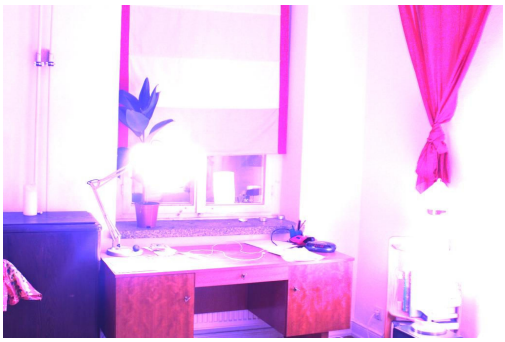
Exponential (δ=0.1)



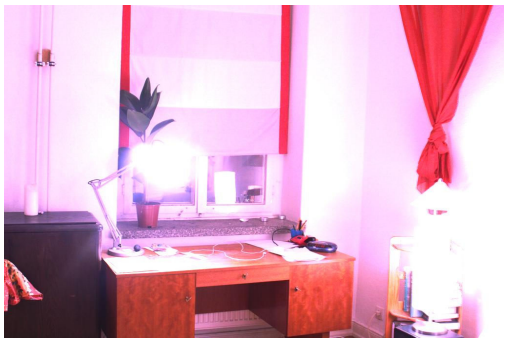
Modified Exponential (δ=0.0001)



Modified Exponential (δ=0.001)



Modified Exponential (δ=0.01)



Modified Exponential (δ=0.1)

Reinhard et al Global Operator

Reinhard et al Local Operator

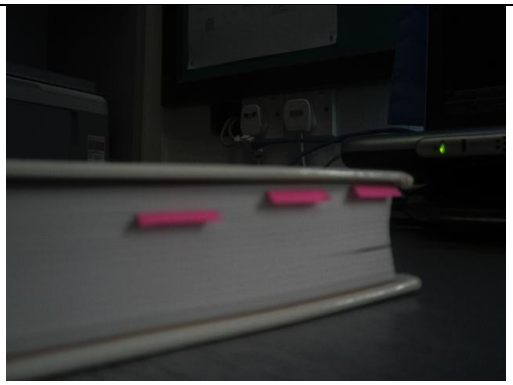
Garrett et al

MATLAB “tonemap”

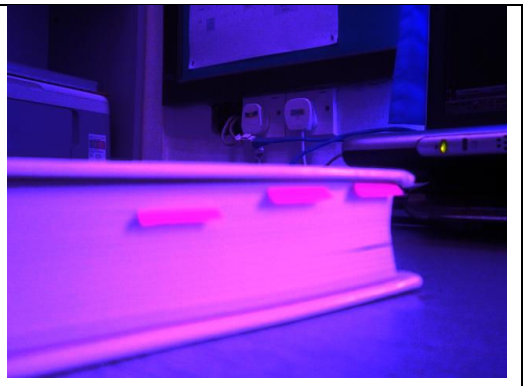
	
Photomatix Details Enhancer	Photomatix Tone Compressor

APPENDIX QQ

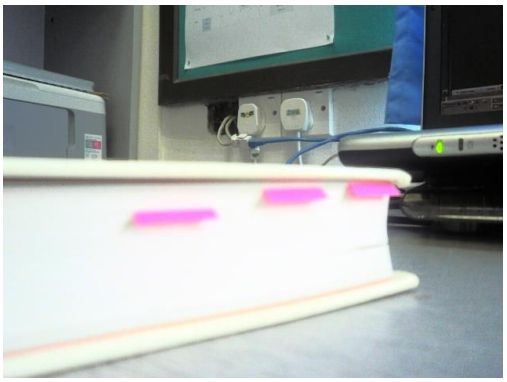
Output Images of Set 4 (book)



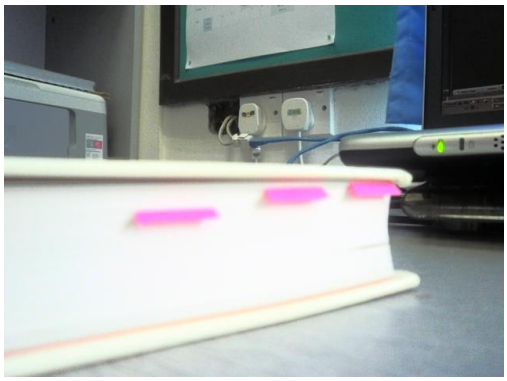
Logarithmic



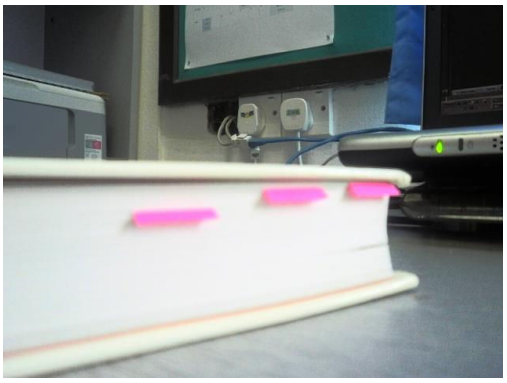
Modified Logarithmic



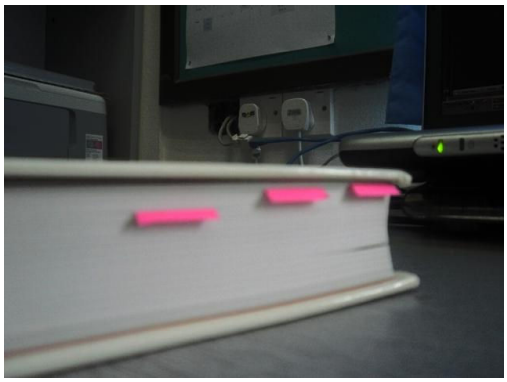
Exponential (δ=0.0001)



Exponential (δ=0.001)

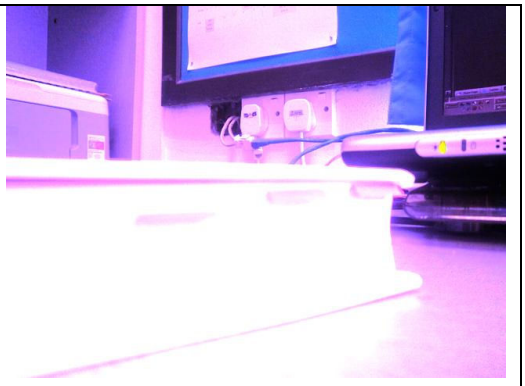


Exponential (δ=0.01)

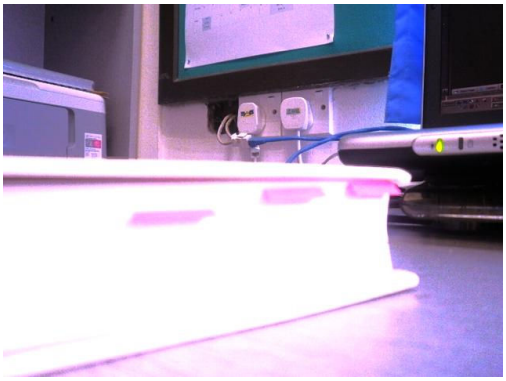


Exponential (δ=0.1)

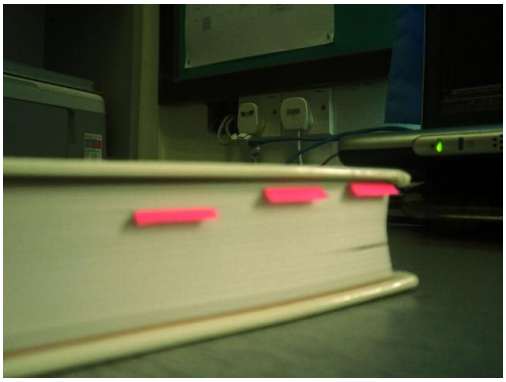
Modified Exponential (δ=0.0001)



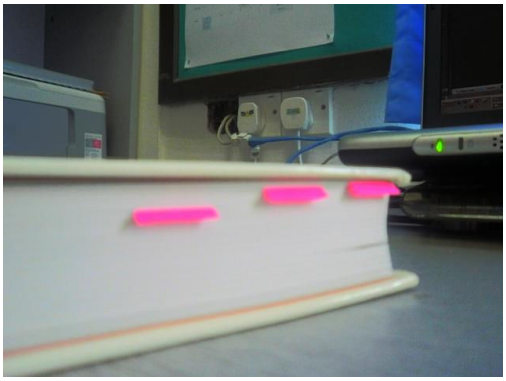
Modified Exponential (δ=0.001)



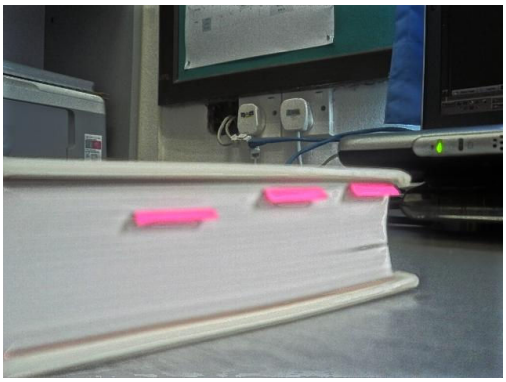
Modified Exponential (δ=0.01)



Modified Exponential (δ=0.1)

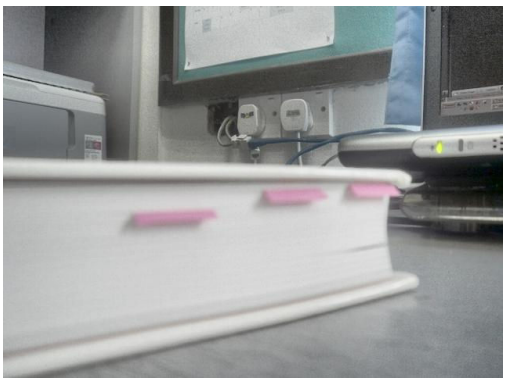


Reinhard et al Global Operator

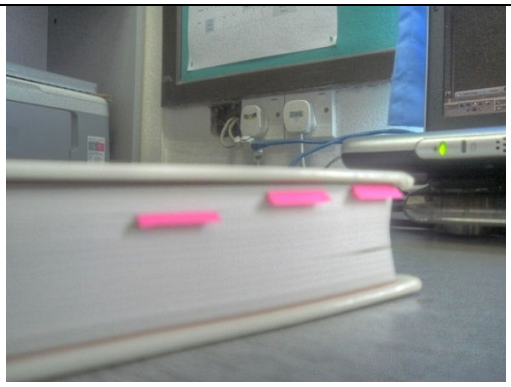
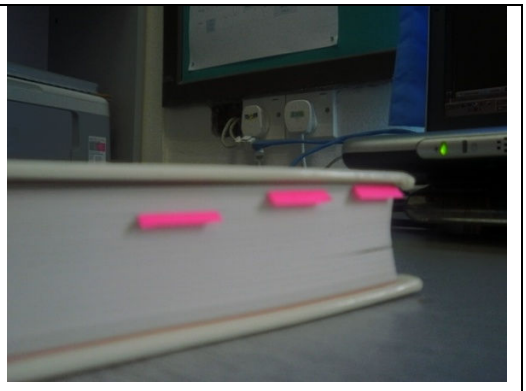


Reinhard et al Local Operator

Garrett et al



MATLAB “tonemap”

	
Photomatix Details Enhancer	Photomatix Tone Compressor

APPENDIX RR

Output Images of Set 4 (lab1)

Logarithmic

Modified Logarithmic

Exponential (δ=0.0001)

Exponential (δ=0.001)

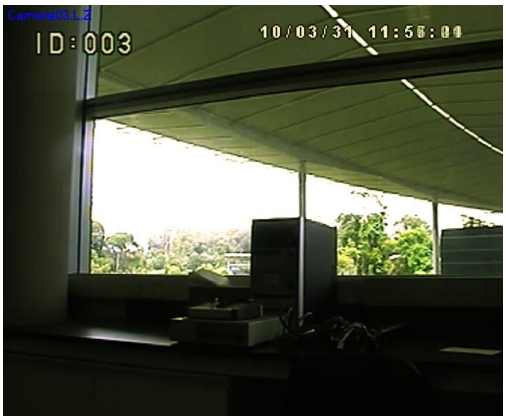
Exponential (δ=0.01)

Exponential (δ=0.1)

Modified Exponential (δ=0.0001)

Modified Exponential (δ=0.001)

Modified Exponential (δ=0.01)



Modified Exponential (δ=0.1)

Reinhard et al Global Operator

Reinhard et al Local Operator

Garrett et al

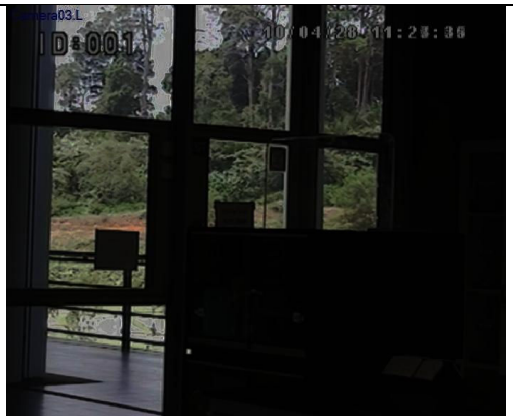
MATLAB “tonemap”

Photomatix Details Enhancer

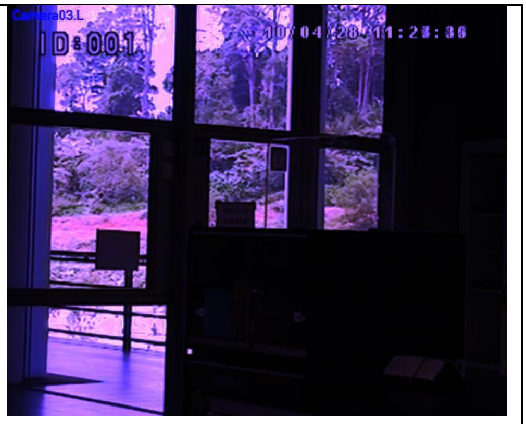
Photomatix Tone Compressor

APPENDIX SS

Output Images of Set 4 (lab2)



Logarithmic



Modified Logarithmic

Exponential (δ=0.0001)

Exponential (δ=0.001)

Exponential (δ=0.01)

Exponential (δ=0.1)

Modified Exponential (δ=0.0001)

Modified Exponential (δ=0.001)

Modified Exponential (δ=0.01)

Modified Exponential (δ=0.1)

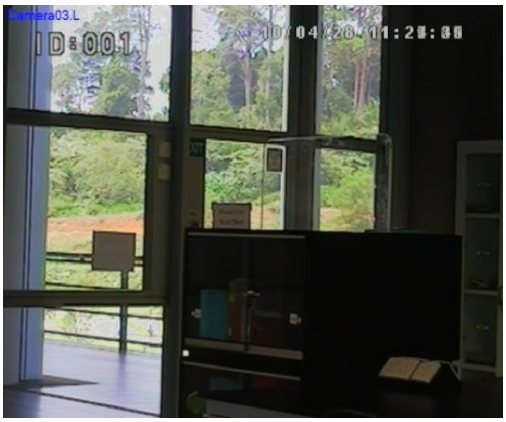
Reinhard et al Global Operator

Reinhard et al Local Operator

Garrett et al

MATLAB “tonemap”

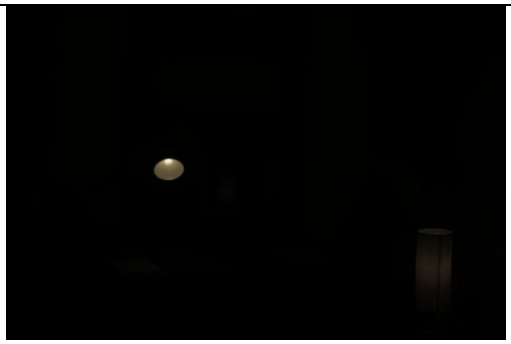
Photomatix Details Enhancer



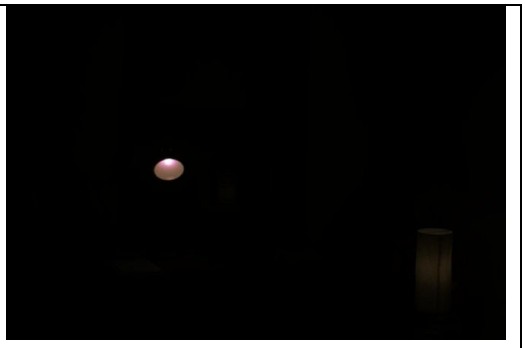
Photomatix Tone Compressor

APPENDIX TT

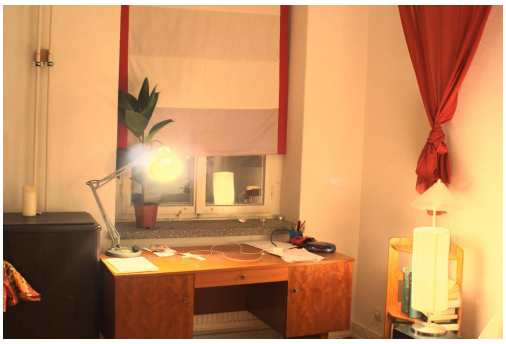
Output Images of Set 4 (window)



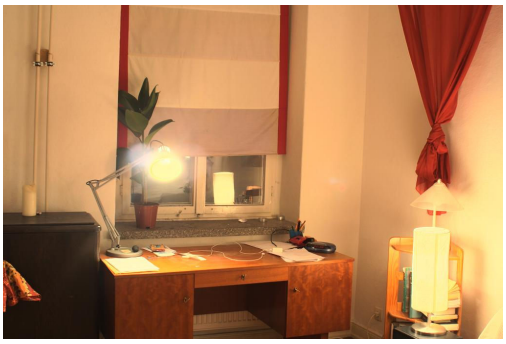
Logarithmic



Modified Logarithmic



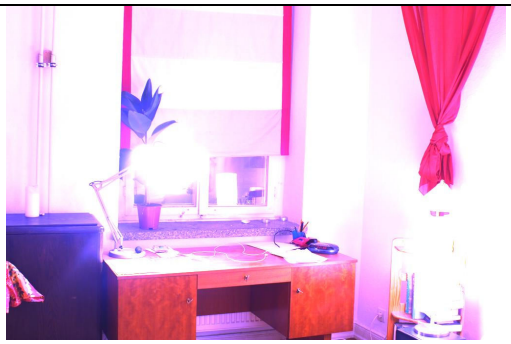
Exponential (δ=0.0001)



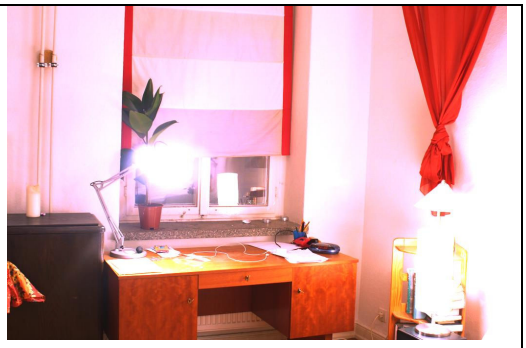
Exponential (δ=0.001)

Exponential (δ=0.01)

Exponential (δ=0.1)



Modified Exponential (δ=0.0001)



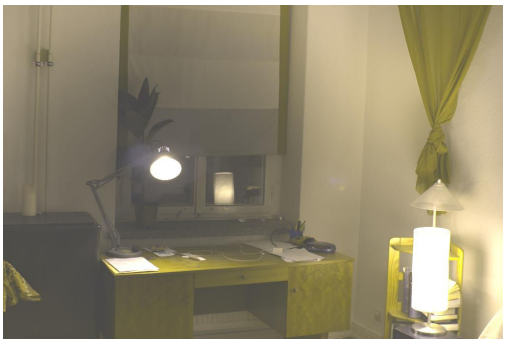
Modified Exponential (δ=0.001)

Modified Exponential (δ=0.01)

Modified Exponential (δ=0.1)

Reinhard et al Global Operator

Reinhard et al Local Operator



Garrett et al

MATLAB “tonemap”

	
Photomatix Details Enhancer	Photomatix Tone Compressor

APPENDIX UU

Computation Time of WDR Image Generation Methods

Computation Time of WDR Image Generation Using MATLAB "makehdr" Function

Input Image	Computation Time (seconds)			Average Computation Time (seconds)
	First Run	Second Run	Third Run	
book	0.2247	0.2168	0.2203	0.2206
lab1	0.2434	0.2443	0.2435	0.2437
lab2	0.4085	0.3905	0.3950	0.3980
window	1.0847	1.0855	1.1212	1.0971

Computation Time of WDR Image Generation Using P. Debevec and J. Malik Method

Input Image	Computation Time (seconds)			Average Computation Time (seconds)
	First Run	Second Run	Third Run	
book	1.8008	1.7641	1.7651	1.7767
lab1	1.7603	1.7380	1.7928	1.7637
lab2	1.0271	1.0190	1.0222	1.0228
window	2.0945	2.1597	2.1760	2.1434

APPENDIX VV

Computation Time of Tone-Mapping Methods

Computation Time of Logarithmic Tone-Mapping

Input Image	Computation Time (seconds)			Average Computation Time (seconds)
	First Run	Second Run	Third Run	
bristolb	1.5704	1.5629	1.5555	1.5629
memorial	0.2807	0.3035	0.2970	0.2937
rossette	0.2417	0.2242	0.2423	0.2361
book	0.3007	0.3014	0.2913	0.2978
lab1	0.5267	0.4857	0.4766	0.4963
lab2	0.4606	0.4483	0.4614	0.4568
window	0.8195	0.8261	0.8263	0.8240

Computation Time of Modified Logarithmic Tone-Mapping

Input Image	Computation Time (seconds)			Average Computation Time (seconds)
	First Run	Second Run	Third Run	
bristolb	2.0307	2.0577	2.0537	2.0474
memorial	0.3422	0.3509	0.3614	0.3515
rossette	0.2932	0.2920	0.2882	0.2911
book	0.3465	0.3555	0.3413	0.3478
lab1	0.5287	0.5326	0.5303	0.5305
lab2	0.5160	0.4973	0.5086	0.5073
window	0.9152	0.8906	0.9139	0.9066

Computation Time of Exponential Tone-Mapping

Input Image	Computation Time (seconds)			Average Computation Time (seconds)
	First Run	Second Run	Third Run	
bristolb	1.5125	1.5084	1.5020	1.5076
memorial	0.2848	0.2700	0.2895	0.2814
rossette	0.2134	0.2171	0.2160	0.2155
book	0.3061	0.2919	0.3027	0.3002
lab1	0.5117	0.4783	0.4787	0.4896
lab2	0.4349	0.4437	0.4618	0.4468
window	0.8075	0.8112	0.8043	0.8077

Computation Time of Modified Exponential Tone-Mapping

Input Image	Computation Time (seconds)			Average Computation Time (seconds)
	First Run	Second Run	Third Run	
bristolb	1.9834	1.9829	1.9371	1.9678
memorial	0.3418	0.3393	0.3487	0.3433
rossette	0.2894	0.2773	0.3015	0.2894
book	0.3521	0.3381	0.3497	0.3466
lab1	0.5274	0.5299	0.5216	0.5263
lab2	0.5059	0.4926	0.4901	0.4962
window	0.8773	0.8604	0.9226	0.8868

Computation Time of Reinhard et al Global Tone-Mapping Operator

Input Image	Computation Time (seconds)			Average Computation Time (seconds)
	First Run	Second Run	Third Run	
bristolb	1.5436	1.5052	1.5408	1.5299
memorial	0.2856	0.3012	0.2798	0.2889
rossette	0.2377	0.2484	0.2214	0.2358
book	0.3070	0.2951	0.2998	0.3006
lab1	0.4825	0.4732	0.4920	0.4826
lab2	0.4395	0.4666	0.4526	0.4529
window	0.7850	0.8080	0.8143	0.8024

Computation Time of Reinhard et al Local Tone-Mapping Operator

Input Image	Computation Time (seconds)			Average Computation Time (seconds)
	First Run	Second Run	Third Run	
bristolb	56.5209	53.3976	57.7739	55.8975
memorial	6.1481	6.1536	6.1047	6.1355
rossette	5.5392	5.4864	5.4881	5.5046
book	4.8829	4.9054	4.8928	4.8937
lab1	6.5932	6.5471	6.6544	6.5982
lab2	6.4403	6.3550	6.4344	6.4099
window	12.4410	12.4180	12.3443	12.4011

Computation Time of Garrett et al Method

Input Image	Computation Time (seconds)			Average Computation Time (seconds)
	First Run	Second Run	Third Run	
bristolb	7.4150	7.4405	7.4321	7.4292
memorial	1.2608	1.2643	1.3184	1.2812
rossette	0.9823	0.9478	0.9307	0.9536
book	0.9701	0.9005	0.9251	0.9319
lab1	1.1223	1.0957	1.0866	1.1015
lab2	1.8972	1.3411	1.3969	1.5451
window	2.2581	2.2855	2.2972	2.2803

Computation Time of MATLAB “tonemap” Tone-Mapping

Input Image	Computation Time (seconds)			Average Computation Time (seconds)
	First Run	Second Run	Third Run	
bristolb	5.6166	5.6338	5.6203	5.6236
memorial	0.8654	0.8123	0.8309	0.8362
rossette	0.8294	0.7513	0.7892	0.7900
book	0.7316	0.7323	0.7383	0.7341
lab1	1.1008	1.0885	1.1640	1.1178
lab2	0.9909	1.0031	0.9909	0.9680
window	1.8553	1.7600	1.7412	1.7855

APPENDIX WW

Result of Observer Evaluation for Set 1

Result of Observer Evaluation for Set 1 (bristolb)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	1	2	2	2	0	5	4.08333	1.88092
Modified Logarithmic	2	4	1	1	2	2	3.25000	1.86474
Exponential ($\delta = 0.0001$)	1	3	2	4	2	0	3.25000	1.28806
Exponential ($\delta = 0.001$)	1	3	2	4	2	0	3.25000	1.28806
Exponential ($\delta = 0.01$)	0	2	3	5	2	0	3.58333	0.99620
Exponential ($\delta = 0.1$)	2	2	3	1	4	0	3.25000	1.54479
Modified Exponential ($\delta = 0.0001$)	1	0	1	3	7	0	4.25000	1.21543
Modified Exponential ($\delta = 0.001$)	0	1	1	4	4	2	4.41667	1.16450
Modified Exponential ($\delta = 0.01$)	0	1	3	2	4	2	4.25000	1.28806
Modified Exponential ($\delta = 0.1$)	2	3	1	1	3	2	3.50000	1.88294
Reinhard et al Global Operator	3	2	3	2	1	1	2.91667	1.62135
Reinhard et al Local Operator	4	2	2	3	1	0	2.58333	1.44338
Garrett et al	0	0	1	5	6	0	4.41667	0.66856
MATLAB "tonemap"	1	0	5	5	1	0	3.41667	0.99620
Photomatix Details Enhancer	5	1	4	1	1	0	2.33333	1.37069
Photomatix Tone Compressor	2	2	2	2	2	2	3.50000	1.78377

Result of Observer Evaluation for Set 1 (memorial)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	0	1	0	4	7	5.41667	0.90034
Modified Logarithmic	0	0	0	0	3	9	5.75000	0.45227
Exponential ($\delta = 0.0001$)	2	6	2	0	2	0	2.50000	1.31426
Exponential ($\delta = 0.001$)	2	4	6	0	0	0	2.33333	0.77850
Exponential ($\delta = 0.01$)	3	7	2	0	0	0	1.91667	0.66856
Exponential ($\delta = 0.1$)	3	1	8	0	0	0	2.41667	0.90034
Modified Exponential ($\delta = 0.0001$)	0	0	1	2	5	4	5.00000	0.95346
Modified Exponential ($\delta = 0.001$)	0	1	1	4	4	2	4.41667	1.16450
Modified Exponential ($\delta = 0.01$)	3	2	2	5	0	0	2.75000	1.28806
Modified Exponential ($\delta = 0.1$)	3	4	0	3	2	0	2.75000	1.54479
Reinhard et al Global Operator	7	2	3	0	0	0	1.66667	0.88763
Reinhard et al Local Operator	6	3	1	1	1	0	2.00000	1.34840
Garrett et al	0	0	0	6	6	0	4.50000	0.52223
MATLAB “tonemap”	5	2	4	1	0	0	2.08333	1.08362
Photomatix Details Enhancer	2	7	3	0	0	0	2.08333	0.66856
Photomatix Tone Compressor	6	5	1	0	0	0	1.58333	0.66856

Result of Observer Evaluation for Set 1 (rosette)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	1	0	0	1	3	7	5.16667	1.46680
Modified Logarithmic	2	1	0	0	3	6	4.58333	2.02073
Exponential ($\delta = 0.0001$)	0	2	4	2	3	1	3.75000	1.28806
Exponential ($\delta = 0.001$)	2	3	1	4	1	1	3.16667	1.58592
Exponential ($\delta = 0.01$)	2	2	4	3	1	0	2.91667	1.24011
Exponential ($\delta = 0.1$)	3	1	1	2	2	3	3.66667	2.01509
Modified Exponential ($\delta = 0.0001$)	0	1	0	3	5	3	4.75000	1.13818
Modified Exponential ($\delta = 0.001$)	1	1	1	5	2	2	4.00000	1.4771
Modified Exponential ($\delta = 0.01$)	1	3	2	3	3	0	3.33333	1.37069
Modified Exponential ($\delta = 0.1$)	2	3	2	1	3	1	3.25000	1.71226
Reinhard et al Global Operator	2	5	2	3	0	0	2.50000	1.08711
Reinhard et al Local Operator	3	0	4	4	1	0	3.00000	1.34840
Garrett et al	0	1	2	3	4	2	4.33333	1.23091
MATLAB “tonemap”	0	0	1	1	5	5	5.16667	0.93744
Photomatix Details Enhancer	3	1	4	2	2	0	2.91667	1.44338
Photomatix Tone Compressor	2	3	3	1	2	1	3.08333	1.62135

APPENDIX XX

Result of Observer Evaluation for Set 2

Result of Observer Evaluation for Set 2 (book)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	0	1	6	5	0	4.33333	0.65134
Modified Logarithmic	0	0	0	0	4	8	5.66667	0.49237
Exponential ($\delta = 0.0001$)	0	6	4	2	0	0	2.66667	0.77850
Exponential ($\delta = 0.001$)	1	4	1	4	2	0	3.16667	1.33712
Exponential ($\delta = 0.01$)	0	5	2	3	2	0	3.16667	1.94342
Exponential ($\delta = 0.1$)	1	4	3	2	2	0	3.00000	1.27920
Modified Exponential ($\delta = 0.0001$)	0	0	0	0	6	6	5.00000	0.52223
Modified Exponential ($\delta = 0.001$)	0	0	0	1	5	6	5.41667	0.66856
Modified Exponential ($\delta = 0.01$)	0	0	0	1	5	6	5.41667	0.66856
Modified Exponential ($\delta = 0.1$)	0	0	0	1	5	6	5.41667	0.66856
Reinhard et al Global Operator	2	5	3	1	1	0	2.50000	1.16775
Reinhard et al Local Operator	6	2	2	1	1	0	2.08333	1.37895
Garrett et al	1	4	3	3	1	0	2.91667	1.16450
MATLAB "tonemap"	0	2	7	1	2	0	3.25000	0.96531
Photomatix Details Enhancer	1	5	4	1	1	0	2.66667	1.07309
Photomatix Tone Compressor	5	1	1	5	0	0	2.50000	1.44600

Result of Observer Evaluation for Set 2 (lab1)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	0	1	5	4	2	4.58333	0.90034
Modified Logarithmic	0	0	0	0	5	7	5.58333	0.51493
Exponential ($\delta = 0.0001$)	0	3	5	3	0	1	3.25000	1.13818
Exponential ($\delta = 0.001$)	0	2	5	4	1	0	3.33333	0.88763
Exponential ($\delta = 0.01$)	1	2	7	1	1	0	2.91667	0.99620
Exponential ($\delta = 0.1$)	0	3	7	2	0	0	2.91667	0.66856
Modified Exponential ($\delta = 0.0001$)	0	1	0	4	3	4	4.75000	1.21543
Modified Exponential ($\delta = 0.001$)	0	0	1	4	3	4	4.83333	1.02986
Modified Exponential ($\delta = 0.01$)	0	0	1	4	3	4	4.83333	1.02986
Modified Exponential ($\delta = 0.1$)	0	0	1	4	3	4	4.83333	1.02986
Reinhard et al Global Operator	2	4	2	3	1	0	2.75000	1.28806
Reinhard et al Local Operator	3	3	2	3	0	1	2.75000	1.54479
Garrett et al	1	7	2	1	1	0	2.50000	1.08711
MATLAB "tonemap"	2	2	5	2	1	0	2.83333	1.19342
Photomatix Details Enhancer	0	1	6	3	2	0	3.50000	0.90453
Photomatix Tone Compressor	2	4	2	2	2	0	2.83333	1.40346

Result of Observer Evaluation for Set 2 (lab2)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	0	0	4	3	5	5.08333	0.90034
Modified Logarithmic	0	0	0	0	4	8	5.66667	0.49237
Exponential ($\delta = 0.0001$)	1	1	5	1	4	0	3.50000	1.31426
Exponential ($\delta = 0.001$)	0	1	7	1	3	0	3.50000	1.0000
Exponential ($\delta = 0.01$)	2	1	5	1	3	0	3.16667	1.40346
Exponential ($\delta = 0.1$)	0	3	5	2	2	0	3.25000	1.05529
Modified Exponential ($\delta = 0.0001$)	0	0	1	2	4	5	5.08333	0.99620
Modified Exponential ($\delta = 0.001$)	0	1	0	2	4	5	5.00000	1.20605
Modified Exponential ($\delta = 0.01$)	0	0	0	3	5	4	5.08333	0.79296
Modified Exponential ($\delta = 0.1$)	0	0	0	4	4	4	5.00000	0.85280
Reinhard et al Global Operator	1	4	3	2	2	0	3.00000	1.27920
Reinhard et al Local Operator	3	3	2	2	1	1	2.83333	1.64225
Garrett et al	0	0	3	5	4	0	4.08333	0.79296
MATLAB “tonemap”	0	0	5	3	4	0	3.91667	0.90034
Photomatix Details Enhancer	0	1	4	3	3	1	3.91667	1.16450
Photomatix Tone Compressor	2	2	5	2	1	0	2.83333	1.19342

Result of Observer Evaluation for Set 2 (window)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	0	0	0	3	9	3.75000	0.45227
Logarithmic (Modified)	0	0	0	0	3	9	5.75000	0.45227
Exponential ($\delta = 0.0001$)	2	5	3	1	1	0	2.50000	1.16775
Exponential ($\delta = 0.001$)	2	4	5	1	0	0	2.41667	0.90034
Exponential ($\delta = 0.01$)	1	7	4	0	0	0	2.25000	0.62158
Exponential ($\delta = 0.1$)	1	3	3	3	1	1	3.25000	1.42223
Exponential (Modified, $\delta = 0.0001$)	0	0	0	3	8	1	4.83333	0.57735
Exponential (Modified, $\delta = 0.001$)	0	0	0	4	7	1	4.75000	0.62158
Exponential (Modified, $\delta = 0.01$)	1	1	5	3	2	0	3.33333	1.15470
Exponential (Modified, $\delta = 0.1$)	0	3	3	4	1	1	3.50000	1.24316
Reinhard et al Global Operator	2	7	3	0	0	0	2.08333	0.66856
Reinhard et al Local Operator	6	4	2	0	0	0	1.66667	0.77850
Garrett et al	0	0	3	6	2	1	4.08333	0.90034
MATLAB (tonemap)	1	2	5	2	1	1	3.25000	1.35680
Photomatix Details Enhancer	0	2	6	3	1	0	3.25000	0.86603
Photomatix Tone Compressor	4	4	2	1	1	0	2.25000	1.28806

APPENDIX YY

Result of Observer Evaluation for Set 3

Result of Observer Evaluation for Set 3 (book)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	1	7	4	0	0	3.25000	0.62158
Modified Logarithmic	0	0	0	4	6	2	4.83333	0.71774
Exponential ($\delta = 0.0001$)	1	7	3	1	0	0	2.33333	0.77850
Exponential ($\delta = 0.001$)	2	4	6	0	0	0	2.33333	0.77850
Exponential ($\delta = 0.01$)	1	6	5	0	0	0	2.33333	0.65134
Exponential ($\delta = 0.1$)	2	6	4	0	0	0	2.16667	0.71774
Modified Exponential ($\delta = 0.0001$)	0	0	1	1	7	3	5.00000	0.85280
Modified Exponential ($\delta = 0.001$)	0	0	0	2	5	5	5.25000	0.75378
Modified Exponential ($\delta = 0.01$)	0	0	1	1	7	3	5.00000	0.85280
Modified Exponential ($\delta = 0.1$)	0	0	0	2	7	3	5.08333	0.66856
Reinhard et al Global Operator	2	5	3	2	0	0	2.41667	0.99620
Reinhard et al Local Operator	2	5	3	2	0	0	2.41667	0.99620
Garrett et al	1	2	7	1	1	0	2.91667	0.99620
MATLAB "tonemap"	0	2	7	3	0	0	3.08333	0.66856
Photomatix Details Enhancer	0	3	8	1	0	0	2.83333	0.57735
Photomatix Tone Compressor	1	6	5	0	0	0	2.33333	0.65134

Result of Observer Evaluation for Set 3 (lab1)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	0	2	3	2	5	4.83333	1.19342
Modified Logarithmic	0	0	1	3	2	6	5.08333	1.08362
Exponential ($\delta = 0.0001$)	0	0	1	4	1	6	5.00000	1.12815
Exponential ($\delta = 0.001$)	0	0	1	3	3	5	5.00000	1.04447
Exponential ($\delta = 0.01$)	0	0	1	4	2	5	4.91667	1.08362
Exponential ($\delta = 0.1$)	0	0	1	4	3	4	4.83333	1.02986
Modified Exponential ($\delta = 0.0001$)	0	0	1	2	2	7	5.25000	1.05529
Modified Exponential ($\delta = 0.001$)	0	0	1	1	3	7	5.33333	0.98473
Modified Exponential ($\delta = 0.01$)	0	0	2	1	3	6	5.08333	1.16450
Modified Exponential ($\delta = 0.1$)	0	0	1	2	3	6	5.16667	1.02986
Reinhard et al Global Operator	0	0	2	2	4	4	4.83333	1.11464
Reinhard et al Local Operator	0	0	2	2	4	4	4.83333	1.11464
Garrett et al	0	0	0	1	2	9	5.66667	0.65134
MATLAB "tonemap"	1	0	3	3	5	0	3.91667	1.24011
Photomatix Details Enhancer	0	1	4	5	1	1	3.75000	1.05529
Photomatix Tone Compressor	0	1	2	2	4	3	4.50000	1.31426

Result of Observer Evaluation for Set 3 (lab2)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	1	3	5	2	0	1	3.00000	1.27920
Modified Logarithmic	0	1	1	7	2	1	4.08333	0.99620
Exponential ($\delta = 0.0001$)	0	1	2	4	5	0	4.08333	0.99620
Exponential ($\delta = 0.001$)	0	1	3	3	5	0	4.00000	1.04447
Exponential ($\delta = 0.01$)	0	1	4	3	4	0	3.83333	1.02986
Exponential ($\delta = 0.1$)	0	1	4	3	3	1	3.91667	1.16450
Modified Exponential ($\delta = 0.0001$)	0	0	1	1	6	4	5.08333	0.90034
Modified Exponential ($\delta = 0.001$)	0	0	1	1	7	3	5.00000	0.85280
Modified Exponential ($\delta = 0.01$)	0	0	2	1	6	3	4.83333	1.02986
Modified Exponential ($\delta = 0.1$)	0	0	2	1	6	3	4.83333	1.02986
Reinhard et al Global Operator	0	1	3	4	4	0	3.91667	0.99620
Reinhard et al Local Operator	0	1	4	3	3	1	3.91667	1.16450
Garrett et al	0	0	2	0	2	8	5.33333	1.15470
MATLAB “tonemap”	0	2	3	4	2	1	3.75000	1.21543
Photomatix Details Enhancer	1	4	5	1	1	0	2.75000	1.05529
Photomatix Tone Compressor	0	3	7	1	0	1	3.08333	1.08362

Result of Observer Evaluation for Set 3 (window)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	0	5	5	1	1	3.83333	0.93744
Modified Logarithmic	0	2	4	4	1	1	3.58333	1.16450
Exponential ($\delta = 0.0001$)	5	6	1	0	0	0	1.66667	0.65134
Exponential ($\delta = 0.001$)	5	6	1	0	0	0	1.66667	0.65134
Exponential ($\delta = 0.01$)	6	5	1	0	0	0	1.58333	0.66856
Exponential ($\delta = 0.1$)	6	5	1	0	0	0	1.58333	0.66856
Modified Exponential ($\delta = 0.0001$)	0	2	0	4	1	5	4.58333	1.50504
Modified Exponential ($\delta = 0.001$)	0	1	1	4	1	5	4.66667	1.37069
Modified Exponential ($\delta = 0.01$)	0	2	0	4	4	2	4.33333	1.30268
Modified Exponential ($\delta = 0.1$)	1	1	3	4	3	0	3.58333	1.24011
Reinhard et al Global Operator	7	4	1	0	0	0	1.50000	0.67420
Reinhard et al Local Operator	4	4	3	0	1	0	2.16667	1.19342
Garrett et al	1	1	4	6	0	0	3.25000	0.96531
MATLAB “tonemap”	1	2	4	5	0	0	3.08333	0.99620
Photomatix Details Enhancer	2	2	5	3	0	0	2.75000	1.05529
Photomatix Tone Compressor	6	4	2	0	0	0	1.66667	0.77850

APPENDIX ZZ

Result of Observer Evaluation for Set 4

Result of Observer Evaluation for Set 4 (book)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	1	3	7	1	0	5.83333	1.58592
Modified Logarithmic	0	1	2	5	4	0	4.00000	0.95346
Exponential ($\delta = 0.0001$)	0	6	4	2	0	0	2.66667	0.77850
Exponential ($\delta = 0.001$)	0	7	4	1	0	0	2.50000	0.67420
Exponential ($\delta = 0.01$)	4	4	4	0	0	0	2.00000	0.85280
Exponential ($\delta = 0.1$)	0	5	3	4	0	0	2.91667	0.90034
Modified Exponential ($\delta = 0.0001$)	0	0	1	1	5	5	5.16667	0.93744
Modified Exponential ($\delta = 0.001$)	0	0	1	1	4	6	5.25000	0.96531
Modified Exponential ($\delta = 0.01$)	0	1	3	7	1	0	3.66667	0.77850
Modified Exponential ($\delta = 0.1$)	0	6	3	2	1	0	2.83333	1.02986
Reinhard et al Global Operator	3	6	3	0	0	0	2.00000	0.73855
Reinhard et al Local Operator	4	7	1	0	0	0	1.75000	0.62158
Garrett et al	0	6	5	1	0	0	2.58333	0.66856
MATLAB "tonemap"	0	1	8	3	0	0	3.16667	0.57735
Photomatix Details Enhancer	0	5	4	3	0	0	2.83333	0.83485
Photomatix Tone Compressor	1	4	6	1	0	0	2.58333	0.79296

Result of Observer Evaluation for Set 4 (lab1)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	1	2	1	3	5	4.75000	1.42223
Modified Logarithmic	0	0	3	1	4	4	4.75000	1.21543
Exponential ($\delta = 0.0001$)	1	1	7	3	0	0	3.00000	0.85280
Exponential ($\delta = 0.001$)	1	2	6	3	0	0	2.91667	0.90034
Exponential ($\delta = 0.01$)	1	3	6	2	0	0	2.75000	0.86603
Exponential ($\delta = 0.1$)	0	3	4	4	0	1	3.33333	1.15470
Modified Exponential ($\delta = 0.0001$)	0	1	0	2	7	2	5.91667	1.67649
Modified Exponential ($\delta = 0.001$)	0	1	0	2	7	2	4.75000	1.05529
Modified Exponential ($\delta = 0.01$)	0	1	3	5	3	0	3.83333	0.93744
Modified Exponential ($\delta = 0.1$)	0	3	3	5	1	0	3.33333	0.98473
Reinhard et al Global Operator	1	6	5	0	0	0	2.33333	0.65134
Reinhard et al Local Operator	1	6	5	0	0	0	2.33333	0.65134
Garrett et al	0	1	2	5	3	1	4.08333	1.08362
MATLAB "tonemap"	0	2	3	3	4	0	3.75000	1.13818
Photomatix Details Enhancer	1	2	6	2	1	0	3.00000	1.04447
Photomatix Tone Compressor	2	3	3	1	3	0	3.00000	1.47710

Result of Observer Evaluation for Set 4 (lab2)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	0	2	1	2	7	5.16667	1.19342
Modified Logarithmic	0	0	1	1	4	6	5.25000	0.96531
Exponential ($\delta = 0.0001$)	0	5	3	4	0	0	2.91667	0.90034
Exponential ($\delta = 0.001$)	0	7	3	2	0	0	2.58333	0.79296
Exponential ($\delta = 0.01$)	0	5	2	5	0	0	3.00000	0.95346
Exponential ($\delta = 0.1$)	1	1	5	4	0	1	3.33333	1.23091
Modified Exponential ($\delta = 0.0001$)	0	0	1	4	5	2	4.66667	0.88763
Modified Exponential ($\delta = 0.001$)	0	0	2	3	5	2	4.58333	0.99620
Modified Exponential ($\delta = 0.01$)	1	3	4	3	1	0	3.00000	1.12815
Modified Exponential ($\delta = 0.1$)	1	1	5	3	1	1	3.41667	1.31137
Reinhard et al Global Operator	2	3	7	0	0	0	2.41667	0.79296
Reinhard et al Local Operator	3	5	4	0	0	0	2.08333	0.79296
Garrett et al	0	1	4	4	3	0	3.75000	0.96531
MATLAB “tonemap”	0	3	3	3	2	1	3.58333	1.31137
Photomatix Details Enhancer	0	2	6	3	0	1	3.33333	1.07309
Photomatix Tone Compressor	1	2	7	0	1	1	3.08333	1.31137

Result of Observer Evaluation for Set 4 (window)

Method	Rating Count						Average Rating	Standard Deviation
	1	2	3	4	5	6		
Logarithmic	0	0	0	0	1	11	5.91667	0.28868
Modified Logarithmic	0	0	0	0	1	11	5.91667	0.28868
Exponential ($\delta = 0.0001$)	3	9	0	0	0	0	1.75000	0.45227
Exponential ($\delta = 0.001$)	4	8	0	0	0	0	1.66667	0.49237
Exponential ($\delta = 0.01$)	3	5	3	1	0	0	2.16667	0.93744
Exponential ($\delta = 0.1$)	0	0	4	6	1	1	3.91667	0.90034
Modified Exponential ($\delta = 0.0001$)	0	2	1	3	6	0	4.08333	1.16450
Modified Exponential ($\delta = 0.001$)	1	2	6	1	2	0	3.08333	1.16450
Modified Exponential ($\delta = 0.01$)	6	1	4	1	0	0	2.00000	1.12815
Modified Exponential ($\delta = 0.1$)	0	0	2	3	6	1	4.50000	0.90453
Reinhard et al Global Operator	4	6	1	1	0	0	1.91667	0.90034
Reinhard et al Local Operator	0	7	5	0	0	0	2.41667	0.51493
Garrett et al	0	1	4	4	3	0	3.75000	0.96531
MATLAB "tonemap"	0	2	5	3	2	0	3.41667	0.99620
Photomatix Details Enhancer	1	5	4	2	0	0	2.58333	0.90034
Photomatix Tone Compressor	4	3	4	1	0	0	2.16667	1.02986